UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

On The Provenance Of Digital Images

Quoc-Tin Phan

Advisor

Prof. Giulia Boato

Co-Advisor

Prof. Francesco G. B. De Natale

Università degli Studi di Trento

April 2019

Abstract

Digital images are becoming the most commonly used multimedia data nowadays thanks to the massive manufacturing of cheap acquisition devices coupled with the unprecedented popularity of Online Social Networks. As two sides of a coin, massive use of digital images triggers the development of user-friendly editing tools and intelligent techniques that violate image authenticity. By this respect, digital images are less and less trustable as they are easily modified not only by experts or researchers, but also by unexperienced users. It has been also witnessed that malicious use of images has tremendous impact on human perception as well as system reliability. Those concerns highlight the importance to verify image authenticity.

In practice, digital images are created, manipulated, and diffused world-wide via many channels. Simply answering to the question "Is an image authentic?" appears insufficient. Further steps aiming at understanding the provenance of images with respect to acquisition device or distributed platforms as well as the processing history have to be considered significant.

This doctoral study contributes solutions to recover digital image provenance under multiple aspects: i) image acquisition device, ii) social network origin, and iii) source-target disambiguation in image copy-move forgery.

Acknowledgement

"No duty is more urgent than that of returning thanks." - James Allen

I am approaching to the end of a journey in my life. Everything seems compact and expressible within about one hundred pages. Nevertheless, my Ph.D. time was more plentiful, and I would like to take this opportunity to express my gratitude to those whose contributions are highly appreciated.

At first and most importantly, my deepest gratitude is sent to Prof. Giulia Boato whose wise supervisory and continuous support played the key role in my research. I have encountered so many obstacles in the beginning of the Ph.D., and would not have been capable to manage without your guidance. I sincerely appreciate your patience and kindness in the position of an advisor.

I would like to give a special thankfulness to my co-advisor, Prof. Francesco G. B. De Natale who always willingly provided thoughtful advice and recommendations for my activities.

I am grateful for the working environment and joyful moments that MMLab members have built for many years. I have to admit a feeling of luck when being part of the group.

A special mention goes to my former advisor Dinh Thuc Nguyen, and Duc Tien Dang Nguyen, who have initialized a preliminary step to fulfill my ambition. Your sincere attitude and generosity are truly honourable.

Thank all of my friends, especially Minh Nhut, Minh Thu, Hai Van, Ngoc Lam, Hong Thuy. Your standing by my side encouraged me to go through hard days in the last three years.

Special thankful feelings are dedicated to Emanuele Sansone, as a friend and as a colleague. I will remember our enjoying moments and thoughtful discussions.

Last but not least, I place the gratefulness to my family in the temple of my heart. Your unconditional love defines me. I am here because of you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Thanks to the massive manufacturing of acquisition devices, the high computing power of computers, photography becomes unprecedentedly advanced. Unfortunately, photography nowadays is losing its innocence. "One Picture is Worth a Thousand Words", an English language-idiom, implies the trustworthiness of traditional photographs, yet a similar implication fails on modern digital images due to the ease of manipulation [4]. The availability of sophisticated photo-editing software, e.g. Adobe Photoshop, Adobe Lightroom, GIMP, CorelDRAW, enables the prevalence of manipulated digital images. Recent advances in Artificial Intelligence (AI) allow to generate plausible image content in autonomous way, deceiving viewers easily. By the popularity of online social networks nowadays, world-wide users have a shared channel for the diffusion of user-generated content, including most common means of multimedia data. When such technologies are mutually available, and if abused for malicious purposes, the negative consequences could be hardly measurable.

Concerns on integrity and authenticity, therefore, motivate effective and efficient techniques to detect or to verify digital images. In the last decade, we have witnessed a set of invented tools easing the analysis at *the current stage* of the image, while the process of understanding on how that image comes to the current stage receives less attention. The latter particular aspect is attributed to *image provenance analysis*. "Provenance" comes from French word, *provenir*, is used popularly in art to prove the source or origin of artwork. In computing, we have found thevery similar implication but with broader sense. Data provenance can be defined as the history of a data item from the time of its creation to the present [5]. Similarly to multimedia data, a digital image also has its own history, of which a full profiling involves not only the creation, or origin, but also the manipulation processes. By this regard, image provenance could imply physical capturing device, or computer generating software that produces the image at the very first stage, or intermediate platforms that produce the new image at intermediate stages. Over the

circulating process, the image might undergo a set of manipulations, before arriving to the current stage.

1.1 Contributions of this study

This doctoral study presents a number of forensic solutions contributing to image provenance analysis including:

Clustering Images by Acquisition Device. Realistic images are created through a capturing device which is tightly associated to an identity. Being able to extract sort of camera fingerprint from images help establish reliable mappings between images and person. As an example, Facebook has filed a patent on camera fingerprinting and its applications[1]. The patent describes how to extract fingerprint of the acquisition camera based only on uploaded images. Based on camera fingerprinting features, Facebook can identify multiple situations in which a user might borrow or change the camera, a user might have multiple social profiles, a user might have real-life relationship with other users and so on. In this situation, uploaded images are unsourced (unknown origin). Occasionally, the information of device can be extracted from the attached metadata, e.g., the Exif header. However, this may be unavailable or can be easily modified or swept out even by non-experts.

It is more practical to consider a situation referred to as *blind*, in which the forensic investigator only has access to pixel values, and no other auxiliary information. With this respect, clustering images with respect to their acquisition devices is a preliminary step. Further steps, such as detecting how many cameras a suspect owns, or how likely an image was taken by the suspect's camera, can derive from clustering results. Indexing images by source camera also results in direct applications in large-scale image retrieval. Towards this goal, this doctoral study introduces a novel approach to cluster images by their acquisition devices in both medium-scale and large-scale contexts. The proposed framework exploits linear dependencies of Sensor Pattern Noise in their intrinsic vector subspaces which better represent relationship of images with respect to acquisition device. Extensive experiments have been conducted to corroborate the accuracy as well as scalability of the proposed framework.

Identifying Social Network Origin of Digital Images. Social networks include both public social networks such as Facebook, Twitter, Flickr, etc., and instant messaging apps like Facebook Messengers, Whatsapp, Telegram, and many others. Such platforms are more and more engaging people in their personal relations taking possession of an important part of their daily life. Huge amount of multimedia contents, mainly photos,

[1] https://patents.google.com/patent/US20150124107

are poured and successively shared on these platforms so quickly that is not possible to follow their paths. This last issue surely grants anonymity and impunity, thus it consequently makes easier to commit crimes such as reputation attack and cyberbullying. In fact, contents published within a restricted group of friends on an instant messaging app can be rapidly delivered and viewed on a public social network by acquaintances and then by strangers without any sort of tracking. In a forensic scenario (e.g., during an investigation), succeeding in understanding this flow could be strategic, thus allowing to reveal all the intermediate steps a certain content has followed.

This study investigates several approaches for tracking the social network origin of digital images. In particular, the inclusion of metadata-based features which are highly distinctive for characterizing the last JPEG compression, and DCT coefficients which could contain traces of multiple JPEG compressions, are analyzed in this study. Moreover, the considered platforms are expanded from public social networks to private communications in instant messaging apps. Not only the last platform where images directly come from, but also the chain of platforms where images have been circulated through are taken into account. The identification of social network origin is cast into classification problem, where traditional classifiers and deep neural networks are proposed and tested. We achieve state-of-the-art performance on broad evaluations.

Source-Target Disambiguation in Copy-Move Forgery. Copy-move forgery is a process of geometrically transforming an indicated region and applying some means of postprocessing within the same image in order to conceal an evidence or to deceive viewers. Copy-move detection methods are able to detect the presence of copy-move forgeries by looking for duplicated regions. Nevertheless, to gain further insights on the image *before* being forged, the source and target regions must be disambiguated. Despite the popularity of copy-move detection methods, source-target disambiguation task has received less attention.

This study proposes a novel solution to solve this forensic problem. The principle idea is based on the fact that color interpolation and postprocessing make copy-move forgery a non-invertible process. It means, the inverse geometrical transformation from target to source introduces flaws. By exploiting this fact, we design a discriminative model by means of a four-stream neural network for this task. We build a data synthesizer to generate a large-scale dataset of forged images enabling the training of such network. We also introduce a method to estimate the transformation matrix from localization mask, broadening the applicability of our method. Despite the fact that this research is on progress, our preliminary results on some benchmarking datasets are highly promising.

1.2 Outline

The remaining sections of this thesis is organized as follows:

In Chapter 2, we provide an overview of popular forensic problems in image provenance analysis, including problems we particularly focus in this dissertation. The first contribution of this doctoral study, a novel method for image clustering with respect to acquisition device, is presented in Chapter 3. We describe the second contribution regarding to the identification of social network origin in Chapter 4. The remaining contribution on source-target disambiguation is presented in Chapter 5. Finally, in Chapter 6 we conclude this dissertation by some remarks.

1.3 List of activities

The presentation of this dissertation follows chronological order of the PhD course.

- We first target the problem of image clustering by source camera in medium-scale contexts, and afterwards extend to large-scale contexts as well. Majority of results presented in Chapter 3 have been published in the following venues:

  Q.-T. Phan, G. Boato, F. G. B. De Natale. Image Clustering by Source Camera via Sparse Representation. In *Proceedings of International Workshop on Multimedia Forensics and Security*, pages 1–5, 2017. [6]

  Q.-T. Phan, G. Boato, F. G. B. De Natale. Accurate and Scalable Image Clustering Based On Sparse Representation of Camera Fingerprint. *IEEE Transactions of Information Forensics and Security*, 2019. [2] [7]

- We study the identification of instant messaging apps, and later expand to public social networks. Partial results presented in Chapter 4 have been published/submitted in the following venues:

  Q.-T. Phan, C. Pasquini, G. Boato, F. G. B. De Natale. Identifying Image Provenance: An Analysis of Mobile Instant Messaging Apps. In *Proceedings of International Workshop on Multimedia Signal Processing*, 2018. [8]

  Q.-T. Phan, G. Boato, I. Amerini, R. Caldelli. Tracking Multiple Image Sharing on Social Networks. To appear in *International Conference on Acoustics, Speech, and Signal Processing*, 2019.

- We study the problem of source-target disambiguation recently and present preliminary results in Chapter 5. This work is still on progress.

[2] Early access: https://ieeexplore.ieee.org/document/8576558

Besides, we have worked on other projects, and published (or submitted) in the following venues:

Q.-T. Phan, M. Vascotto, G. Boato. Hue Modification Localization By Pair Matching. Submitted to *European Signal Processing Conference*, 2019.

E. Sansone, **Q.-T. Phan**, F. G. B. De Natale. Coulomb Autoencoders. Submitted to *International Conference on Machine Learning*, 2019.

F. Lago, **Q.-T. Phan**, G. Boato. Visual and Textual Analysis for Image Trustworthiness Assessment within Online News. Submitted to *Security and Communication Networks*, 2018.

F. Lago, **Q.-T. Phan** and G. Boato. Image Forensics in Online News. In *Proceedings of International Workshop on Multimedia Signal Processing*, 2018.

Q.-T. Phan, D.-T. Dang-Nguyen, G. Boato, F. G. B De Natale. Using LDP-TOP in Video-Based Spoofing Detection. In *Proceedings of International Conference on Image Analysis and Processing*, pages 614–624, 2017.[9]

Q.-T. Phan, A. Budroni, C. Pasquini, F. G. B. De Natale. A Hybrid Approach For Multimedia Use Verification. In *MediaEval*, 2016. [10]

Q.-T. Phan, D.-T. Dang-Nguyen, G. Boato, F. G. B De Natale. Face Spoofing Detection using LDP-TOP. In *Proceedings of International Conference on Image Processing*, pages 404–408, 2016. [11]

Chapter 2

Digital Image Provenance

When you see digital images on the Web, you might ask questions to yourself: *Where do they come from? Do they reflect the reality? If not, how have they been modified?* It is important to answer such questions, particularly in forensic science. Unfortunately, answering them is not easy. By relating to analog forensics, [12] justifies why such questions are difficult. Interactions between two entities in the physical world leave *patterns* (or footprints) allows *unconstrained* forensic investigators to analyze the scene to find even the subtlest traces. Digital forensic investigations, on the other hand, are limited to finite states of computers. All traces can be erased *completely*, defeating even *unconstrained* forensic investigators.

In practice, there neither exist unconstrained forensic investigators, nor unconstrained perpetrator that can erase completely traces. Actions taken on digital data often leave intrinsic evidences, which serve as the basis for research on multimedia forensics moving on. Multimedia forensics have been considered as relatively young, but rapidly received attentions from researchers from different communities. Two original main missions in multimedia forensics are to identify the source device (or sensor), and to reveal traces of manipulation. The first mission is dedicated to answering the first aforementioned question, while the second mission means to answer the second question. Recent efforts in multimedia forensics research not only extend the first question to different contexts: Social Network (SN) origins and image phylogeny, but also answer the third question: how to reconstruct the manipulation process.

This study concentrates more on the first and the third question. By our classification, the focus of the study is therefore dedicated to image provenance analysis which unveils the origin of digital images and the process on which images have been manipulated. To this end, we consider leaf nodes of the hierarchy in Figure 2.1 as elements of image provenance analysis, and attempt to describe concrete problems as well as remarkable solutions in this Chapter.

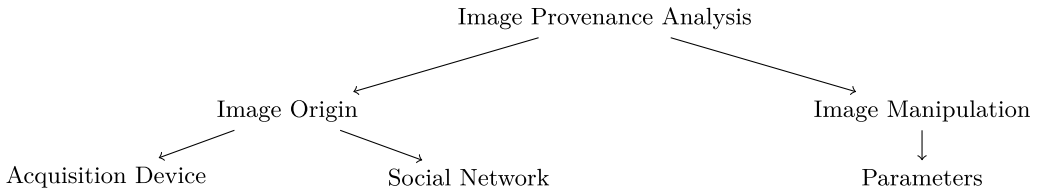


Figure 2.1: Hierarchy of image provenance analysis.

Section 2.1 briefly introduces key problems and solutions regarding to acquisition device, while Section 2.2 discusses the recovery of social network origin. The reconstruction of popular image manipulations is presented in Section 2.3.

2.1 Image Acquisition Device

Images, a popular form of multimedia taken by digital cameras, are faithfully associated with an identity. Being able to retrieve information of the capturing device is essential for forensic applications. In this section, we review remarkable solutions for three well-known problems:

- Source Camera Classification: classifying images according to their origin, given a set of possible devices.
- Source Camera Identification: proving a given image was acquired by a specific device.
- Image Clustering by Source Camera: clustering images by their acquisition device.

The aim of Source Camera Classification (SCC) is to assign a given image into one of possible devices, e.g. Canon versus Kodak, scanner versus digital cameras, etc. The challenge here is to develop a discriminative information of cameras from pixel values. This problem is addressed in the literature by means of *software* and *hardware* artifacts. Software artifacts are caused by particular processing deployed inside the camera. Remarkable artifacts are Color Filter Array (CFA) configuration and demosaicing [13, 14, 15], and color statistics [16, 17]. CFA configuration settles the locations of directly observed pixels and interpolated pixels, while interpolation algorithm decides on how neighboring pixels are correlated. In [13], Expectation Maximization (EM) is applied in a similar way to [18] for finding coefficients of a linear interpolation model, while [14] searches for CFA pattern that minimizes the interpolation error of a linear model estimated by least squares. An extension of [14] is introduced in [15] by considering cross-channel color interpolations. In addition to CFA patterns and interpolation algorithms, artifacts of auto-white balance process are exploited to identify source camera [19].

Despite being effective, methods based software artifacts are sensitive to camera configurations at the capturing time. For this reason, research is oriented to hardware artifacts, such as lens radial distortion, vignetting, lens chromatic aberration.

Most of camera lenses have spherical shape which can distort images, for instance, a straight line deforms to a curved line. This deformation refers to lens radial distortion which is a function of radius from the image center [20]. The estimation of lens radial distortion involves estimating parameters of distortion function. Different lenses inherit different levels of distortion, and thus an estimation of distortion parameters is useful in SCC. Originating from the attempt to estimate chromatic aberration caused as the light of different wavelengths fails to converge at a single point [21], work in [22] exploits chromatic aberration for SCC by estimating color misalignment between red and green channels, as well as blue and green channels, and searches for maximum mutual agreement between two estimates. Brute-force search for color misalignment is computationally expensive, [23] proposes a method to work on sub-sampled blocks to alleviate the complexity. Another optical distortion, vignetting, is known as the decrease of color intensity from the optical centre to the borders of the image, making the image brighter in the centre and darker towards the borders. For classifying different lens models (potentially for SCC problem), [24] estimates parameters formulating the light attenuation pattern of the lens by Maximum Likelihood Estimation (MLE). Recent advances of deep learning, in particular Convolutional Neural Networks (CNNs), allow to extract features and to classify the input image in an end-to-end system [25]. The learnt features have been proved to be discriminative enough for fingerprinting camera models [26].

Techniques for SCC rely on discriminative information of camera brands/models, yet infeasible for identifying particular device. On the other hand, the aim of SCI is to prove that a given image has been taken by a specific device. To this end, researchers seek for fingerprints individualizing a particular device instead. Similarly to human fingerprints or iris, digital cameras are also associated with their fingerprints. Taking this as the motivation, sensor dust of DSLR cameras is served as peculiar footprints. The pixel intensities corresponding to lens dust are modelled as Gaussian and SCI is cast into detecting and matching dust spots [27]. This type of fingerprint is not robust since it is easily confused with highly textural regions. Introduced the first time by [28], Sensor Pattern Noise (SPN) can be considered as "biometrics" of digital cameras. Due to the imperfections of camera sensors, each taken image is overlaid with a noise-like signal called SPN which composes Fixed Pattern Noise (FPN) and Photo-Response Nonuniformity (PRNU). FPN is caused by dark currents, and usually suppressed automatically by digital cameras, while PRNU is the dominant component of SPN caused by different sensitivity of pixels to incoming light. Due to this reason, the terminology SPN and PRNU are sometimes used interchangeably.

In [28], a reference pattern of SPN is estimated by averaging multiple noise residuals extracted from images belonging to a specific camera. Averaging suppresses random components in noise residuals and strengthens SPN. Afterwards, a given image is attributed to a camera if the correlation between its noise residual and camera reference pattern exceeds a predefined threshold. This threshold is found based on Neyman-Pearson approach, in such a way that it minimizes False Rejection Rate (FRR) and maintains False Acceptance Rate (FAR) at an acceptable level. Despite the effectiveness of noise residual averaging, it lacks a theoretical justification. [29] afterwards develops a theory for PRNU estimation, which shows that PRNU is a MLE given the acquisition model. Details of the estimation is provided in Appendix 7.1 for interested readers. Also shown in [29], the variance of the estimated PRNU depends on the number of images, and which type of images used for the estimation. Generally, a good estimate requires the number of images to be large and the luminance of images should be high (yet not saturated). If the camera is available, those conditions can be fulfilled by taking large number of smooth images. Linear patterns such as color interpolation, JPEG blocks, row-wise and column-wise operation of sensors and processing circuits make PRNUs of different cameras slightly correlated. [29] also proposes row-wise and column-wise zero-meaning in addition with Wiener filtering on frequency domain to suppress non-unique artifacts.

Based on the theoretical framework of [29], SCI is equivalent to detecting whether or not the test image contains the PRNU of a specific camera. This problem is formalized as hypothesis testing and the optimal detection statistic turns out to be normalized correlation under the assumption that the random noise component is independent with PRNU component. In [30], Peak to Correlation Energy (PCE) is proposed as improved detection statistic for rescaled and cropped images. PCE can also reduce the effect of periodic patterns in noise residuals and PRNUs [31]. The use of correlation over circular Cross-Correlation Norm (CCN) [32] is another alternative detection statistic in SCI.

Majority of research later aim at improving the reliability of SCI. For instance, [33] proposes six enhancing models to alleviate the contamination of scene details given SPN as input. [32] addresses the same problem as [33] but proposes to extract only phase component of SPN spectrum, called phase SPN. Differently, [34] detects and suppresses peaks in high frequency band of SPN spectrum. Rather than improving SPN with postprocessing, [35] aims at better extracting SPN by using an innovative filter, block-matching and 3D filtering (BM3D) [36]. BM3D groups similar 2D blocks to form 3D arrays and finds sparse representation of 3D arrays on frequency domain. Thanks to the similarity of 2D blocks on pixel domain, the signal and the noise are well separated.

Regarding to Image Clustering by Source Camera (ICSC), which is our main focused problem towards uncovering image origin. This problem is addressed in *blind scenario*

where there is no auxiliary information is available except pixel values. As aforementioned, in order to properly estimate SPN, a number of smooth and uniformly bright images should be collected [29]. Unfortunately, this requirement is usually not fulfilled in a blind scenario, where all images are unlabeled and no assumption can be made about the visual content. Consequently, SPN can just be coarsely approximated from the noise residual of a single image, which contains not only the pattern noise but also various other noise sources, such as shot noise and noise resulting from lossy compression or other filtering. Different methods have been proposed to enhance the SPN estimation and matching, yet require a labeled training set, making them unsuitable for image clustering by source camera in unsupervised scenarios.

Existing unsupervised techniques are typically based on the normalized correlation among SPNs, used as a similarity measure, whose degree of reliability is limited by the impact of multiple noise sources. In [37], an image is assigned to a group if the correlation between its noise residual and the relevant centroid exceeds a threshold, approximated by a quadratic model. Markov Random Fields are applied in [38, 39] to iteratively assign a class label to an image based on the consensus of a small set of SPNs, called membership committee. This raises another problem on how to choose a good committee, especially on asymmetric datasets where cluster cardinalities are unbalanced. In [40, 41, 42], a hierarchical partition - a binary tree containing singleton clusters as leaf nodes and whose root node is a cluster containing all data points - is built by hierarchical clustering. The major problem of existing hierarchical approaches is the sensitivity to noise and outliers, as a wrong assignment might result in the propagation of errors to higher tiers. Multiclass spectral clustering is applied in [43] to partition an undirected graph of unsourced images. The algorithm starts with two clusters and stops when it finds a cluster containing only one member. This stopping condition is heuristic, and as been improved by using normalized cut in [44]. Recently, in [45, 46], multiple base partitions are obtained on top of multiple binarized undirected graphs and then combined to form a complete clustering solution.

Another important problem that has to be taken into account is scalability. In practical applications, often the clustering has to be applied to large databases, containing huge numbers of high-resolution images. To the best of our knowledge, only the method in [2] addresses large-scale clustering of camera fingerprints, where the main idea is to split the dataset into small batches, which can be efficiently loaded on RAM, and to apply a coarse-to-fine clustering.

Remaining issues and our concentration. Acquisition device forensic has become matured, but performance of available techniques is still dependable on critical assumptions. For instance, the accuracy of SCI and ICSC heavily relies on assumption of pixel alignment, which is easily broken in practice. Despite the fact that pixel alignment is just

a mild assumption in SCC, it instead requires knowledge about the complete set of suspected cameras. Moreover, the consideration of large-scale contexts and adversarial-aware contexts are essential. In this study, we address the problem of ICSC by proposing a novel method for accurately clustering images in both medium-scale and large-scale contexts.

2.2 Social Network Origin of Images

If cheap acquisition devices and user-friendly editors allow for the creation of manipulated content, online social networks (SNs) are the privileged channel for their systematic and uncontrolled distribution. Through these services, users are free to upload, share, download and re-upload unprecedented numbers of images, videos, audio tracks, with basically no limitation on the information that can be exchanged within a time frame.

The negative impact of malicious manipulations over such sharing platforms are immense, and witnessed by several cases. After the Malaysia Airlines jetliner carrying 227 passengers and 12 crew vanished on March 2014, online scammers posted on social media manipulated images and videos claiming that the flight had been found, causing severe depressions for families involved in the accident [47]. Moreover, misleading information can be propagated even if images and videos are authentic but artificially inserted in wrong contexts. In all these cases, the information went viral quickly and uncontrolled, creating chaos and panic among people. For this reason, researchers started addressing the wider problem of *multimedia verification*, i.e. the classification of verifiable online multimedia content with respect to its credibility and veracity. In SN context, fake content refers to any publication or post with multimedia content that does not represent accurately the event that it refers to [48]. This was recently one of the main objectives of the European project REVEAL – REVEALing hidden concepts in Social Media [1], which investigated the case of Twitter and proposed solutions to identify fake tweets [49, 50] by jointly analyzing textual, visual and user information.

While image forensics techniques have been employed to solve the overall verification problem, they have been shown to be generally weak for this task, for which textual features are proved to be more informative. We can identify two main reasons for that. First, when uploaded and shared via SN platforms, data undergo strong processing such as compression and resizing, thus substantially removing traces left by previous manipulation and making the detection of tampering extremely difficult [51]. In fact, to optimize the storage, transfer bandwidth as well as display, most SNs enforce their own compression/resizing policy on images and videos, which is generally not published nor fixed. Secondly, image forensic techniques are unable to differentiate between harmful and un-

[1] https://revealproject.eu

harmed forgeries (like text or logo insertion, or image quality enhancement), so that standard forensic features often give inaccurate information and must be complemented by external information retrieved online [50, 52].

Given the pervasive use of SNs, a huge source of multimedia content on the Internet nowadays belong to such platforms. Nevertheless, being uploaded and downloaded to/from multiple platforms, an image is losing its origin. It is important to identify a posteriori the SN platform the image come from. This task can support other forensic tasks such as multimedia content verification, or image phylogeny. SN origin gives hint on how an image has been processed on a SN platform and forensic methods can be adapted accordingly. Towards this goal, researchers exploits mainly artifacts of JPEG compressions for identifying SN origin.

JPEG is an image format that has been chosen by many platforms as default saving format thanks to its great tradeoff between compression rate and visual fidelity. JPEG compression is lossy [53], which means that the compressed signal is different from the original signal. The information loss is due to two phases: Discrete Cosine Transform (DCT) and quantization. In theory, forward DCT and inverse DCT are perfectly revertible, but no physical implementation can obtain perfect accuracy. Forward DCT transformation results in DCT coefficients, which are then quantized and rounded to limit the precision needed for their representations. Quantization, the main cause of information loss, is many-to-one mapping which discards visually insignificant information. Due to the lossy nature, JPEG compression leaves inevitable artifacts on the compressed image.

The pioneering work in [54] identifies SN origin with respect to Facebook, Twitter and Flickr. Although both platforms perform JPEG compression on uploaded images, the chosen compression parameters might be different, resulting peculiar patterns. The approach in [54] analyzes such peculiar patterns in histograms of AC coefficients. Results show that such histograms are distinguishable among three platforms regarding to magnitude and position. The SN origin identification is cast to classification problem. Experimental results are highly accurate, highlighting the feasibility of the problem. Furthermore, the image quality before uploading can be detected if images are not resized by the platform. While such results are limited to only three platforms, [55] expands the analyses on Tumblr, Imgur, Whatsapp, Tinypic, Instagram and Telegram. [55] considers a combination of quantization coefficients of luminance and chrominance channels, number of metadata entries found in header, number of JPEG markers, as peculiar patterns. In fact, there is a correlation between coefficients of quantization tables used in [55] and histogram of DCT coefficients in [54]. The key difference is that if an image is uploaded on a platform, downloaded, and re-uploaded to another platform, the method in [55] cannot detect the first platform because the considered peculiar patterns belong only to the last

platform. This scenario is referred to as *multiple sharing*, and will be analyzed in more detail in this study. Apart from traditional classifiers, [56] shows the potential use of CNNs in the detection of SN origin on small 64×64 image patches. It is not surprising when the CNN-based approach in [56] achieves state-of-the-art performance due to highly representational power of deep networks.

Remaining issues and our concentration. Despite the distinctiveness of JPEG compression artifacts, a thorough exploration of multiple sharing scenarios remains open. In this study, we build large-scale benchmarking datasets to conduct such analyses, both on public social networks and instant messaging apps, and propose innovative methods to improve identification performance.

2.3 Manipulation Reconstruction

Image manipulations are operations resulting pixel values that deviate from typical camera outputs. Under this general sense, image manipulation reconstruction refers to forensic techniques that trace back how the given image has been manipulated. We found this definition is close to recovery of the processing history mentioned in [57], and estimation of manipulation parameters in [58, 59]. Being able to reconstruct image manipulations is useful to build digital image provenance in legal cases, or to provide prior knowledge supporting other forensic tasks.

Although image manipulations in reality can be highly complex and divergent, vast studies have been focused on specific manipulations as listed in [57]: contrast enhancement (and gamma correction), image resizing, median filtering, copy-move forgery, and JPEG compression. In addition to such well-studied operations, hue modification has been addressed as well by [60, 61]. Besides detection of such manipulations, i.e. "whether or not they were applied", efforts have been acknowledged to answer to "how they have been done". In the followings, we present principle ideas of these studies.

Contrast enhancement and gamma correction. Contrast enhancement and gamma correction are *non-linear* mappings of pixel values, followed by quantization. These operations result in three main kinds of artifacts:

- Correlations on image bispectrum.
- Artifacts (peaks and empty bins) on color histogram.
- Periodicity change on histogram of DCT coefficients if the image has been JPEG compressed once.

Non-linear gamma correction creates higher-order correlations of the signal in frequency domain. By analyzing bispectrum of the image, the parameter gamma is estimated as the

one to which the inverse of the corresponding gamma correction yields minimal normalized bispectrum [62]. The image is decomposed into 1D signals by horizontal scans, and the estimates from 1D signals are averaged to obtain the estimate gamma. More generally, if both of contrast enhancement and gamma correction are fairly mapping functions (non-parametric), they results in peaks and empty bins on color histograms because of many-to-one mappings. These artifacts are clearer when comparing with histograms of natural images, which are generally smooth. To smooth histograms, a bin value can be well interpolated by all other bin values. A peak on the histogram of contrast enhanced image therefore indicates many-to-one mapping, causing a significant interpolation error. Under a probabilistic model of errors, peaks as well as the set of pixel values before mapping are gradually identified in iterative fashion [63]. In the end, both of mapping functions and histogram of the unaltered image are jointly estimated. More interestingly, despite the non-linearity of contrast enhancement and gamma correction, histogram of the image before and after being manipulated have linear relationship. Using this fact, [64, 65] solve an optimization problem alternatively for recovering the mapping and the histogram of unaltered image. [66] targets a rather particular case where double JPEG compression is interleaved by a *linear* contrast enhancement. The estimation of contrast enhancement in this case becomes finding the slope of the linear mapping. Due to the linearity of DCT, if the pixel values are scaled by a factor, it leads to the scaling of DCT coefficients by the same factor. In theory, the periodicity on the histogram of DCT coefficients of a doubly compressed image is scaled by that factor as well. The estimation of the slope (the scaling factor) therefore can be performed via exhaustive search such that it minimizes the distance between empirical and theoretical periodicity.

Image resizing. Regarding to the problem of resizing factor estimation, prior art considers three main artifacts:

- Periodicity of the variance of second-order derivative of columns and rows.
- Dependency of samples to their neighbors.
- Periodic patterns of JPEG blocking artifacts.

The first artifact is exploited by [67] to detect interpolated signal and to estimate the resampling factor. Based on the observation that the variance of the second-order derivative of an interpolated signal has a periodicity equal to the resampling factor. The second derivative signal of each row are therefore computed and their absolute values are averaged over all rows to obtain a second-order derivative trace. Since this trace is periodic, frequency analysis can facilitate interpolation detection and resampling factor estimation. The resampling factor is assumed to be at least two; otherwise aliasing will introduce sort

of ambiguity in the estimation. Another widely used method to detect the presence of resampling in images in the literature is [68], and this method leverages the second mentioned resizing artifact. When a signal is resampled, the interpolation results in the set of samples are correlated in the *same* way to their neighbors. [68] applies EM to jointly estimate the probability that a sample is correlated to their neighbor (forming a p-map), and interpolation coefficients. The method can detect the presence of resampling by analyzing peaks in the spectrum of p-map. Nevertheless, the exact parameters of resampling, i.e. resizing factors, cannot be recovered as different resamplings might result in similar interpolation coefficients. [69] further explains the spectral representation of p-map and proposes an accelerated and simplified method to detect resampling. This effort offers a hint to perform brute-force matching using p-map spectrums in order to recover resizing factor. The last resizing artifact is considered by [70] when double JPEG compression is interleaved by resizing. As a consequence, periodic patterns introduced by JPEG blocking artifacts after resizing can be analyzed on frequency domain to recover a set of candidate resizing factors. Without basing on any prior knowledge of three aforementioned artifacts, [71] trains deep neural networks (DNNs) on images resized by a finite set of resizing factors, i.e. 0.5 to 1.5, step 0.1. While the detection of resampling is possible, what remains open is to overcome aliasing and estimate fine-grained resizing factors.

Median filtering. Median filtering is known as a common denoising technique for impulsive noise removal. It operates using a sliding window called kernel and the median pixel value is selected at every window position. The single parameter of this technique is the kernel size. Although many detectors have been proposed for detecting median filtering, the estimation of its kernel size draws less attention. Techniques for median filtering detection typically focus on statistical properties of the first-order difference image. It is pretty obvious that median filtering with different kernel sizes leave different statistical traces, and since the number of practical kernel sizes are limited, typical classifiers can be trained on statistical features like histograms of first-order difference image for finding kernel size. Indeed, [72] confirms this capability. In particular, given the image under investigation, [72] first proposes to extract an image residual, which is the difference of median filtered image and the image itself in order to mitigate the interference of scene details. After that, coefficients of a linear predictor are learnt from the image residual, and used as features for estimating kernel size. Different from model-based approach, [58] exploits DNNs for this task and obtains promising results. One characteristic of this network is the inclusion of front constraint filters which extract image residuals. While traces of median filtering are inevitable, they are also easily erased by counter-forensic methods [73, 74].

JPEG compression. A common processing which has been deploying into popular

imaging devices nowadays is JPEG compression. The parameters of JPEG compressions are stored in JPEG file header, and it is often pointless to spend efforts on reconstructing them. It is useful, however, to detect the presence as well as parameters of JPEG compression when a JPEG compressed image is decompressed and resaved in a format different from JPEG, e.g. analyzing blocking artifacts [75] or the distribution of most significant digits of AC DCT coefficients [76], to detect whether or not an image has been JPEG compressed *once*. In particular, the DCT coefficients are reasonably assumed as Gaussian distribution and quantization steps can be estimated using MLE [75]. It is also straightforward to estimate the quality factor by fairly assuming that the quantization table used is a scaled version of the standard one. When the image is doubly JPEG compressed with grid alignment, the distribution of DCT coefficients changes depending on the interplay of primary and secondary quantization steps. Based on this fact, the primary quantization steps and if necessary, the primary quality factor can be estimated. A primitive method towards this purpose creates simulated histograms of doubly compressed image and matches with the observed histogram [77]. This approach is computationally expensive and can be replaced by training a set of neural networks for each of low spatial frequencies to predict the primary quantization steps [77]. SVM classifier has been adopted in [78] for this purpose. Particularly challenging cases such as double JPEG compression interleaved by resizing or non-aligned double JPEG compression have been addressed by [70, 79].

Hue modification. In digital image forensics, hue modification has not been well-known despite its popularity in practice. First, hue modification is defined as the change in the ratio of red, blue and green channels [60]. For geometrical interpretation, hue modification is the rotation of a vector with three color components, i.e. red, green, blue by a specific angle. The estimation of angle has been firstly addressed by [60] based on CFA configuration. In principle, given CFA configuration, one can estimate the number of interpolated pixels as those are often smaller than the maximum and larger than the minimum of their neighbors. At observable cells with respect to CFA configuration, this count is notably smaller than those at unobserved positions. Hue modification violates this property. Based on the fact that hue modification is periodic, i.e. modifying by an angle and subsequently by its complementary angle returns back the unmodified image, the estimation of hue modification can be done by searching over a finite set of angles and verifying the aforementioned property. By assuming the camera PRNU can be estimated, [61] proposes to first modify the PRNU pattern by a finite set of possible angles and afterwards match with noise residual of the questioned image.

Copy-move forgery. The common principle of copy-move detection methods is to find duplication in the forged image. The closer the two regions are, the easier copy-

move detection methods can detect. There is, however, a complementary problem after copy-move detection. It is the disambiguation of source and target. The essential of this problem is undeniable if a forensic investigator desires to understand the image before being forged. With this respect, [80] builds a large-scale synthetic dataset and designs an end-to-end CNN, called BusterNet to jointly localize duplicated regions and to discern source and target. Due to this dual task, BusterNet composes two sub-networks, one is trained for localizing duplication and one is trained for localizing target. The outputs of two sub-networks are combined to come up with final decision. Despite interesting motivation, discernibility of BusterNet needs to be revisited, in the sense that it is fairly weak in localizing duplicated regions. While BusterNet requires fixed input size, images are typically resized before feeding to the network without respecting image aspect ratio. This technical issue limits BusterNet in practical considerations.

Remaining issues and our concentration. Being part on the line of manipulation reconstruction, this study addresses the problem of source-target disambiguation in copy-move forgery. Despite its importance, this problem has been overlooked and just been tackled recently by [80]. As discussed, [80] exhibits several weaknesses and indeed its discernibility is pretty weak. We will describe how the proposed solution can solve this problem.

Chapter 3

Clustering Images by Acquisition Device

Clustering images according to their acquisition devices is typically faced by means of camera Sensor Pattern Noise (SPN). Such a problem is challenging since SPN is a noise-like signal, hard to be estimated and easy to be attenuated or destroyed by many factors. Moreover, the high dimensionality of SPN hinders large-scale applications. Existing approaches are typically based on the correlation among SPNs in the pixel domain, which might not be able to capture intrinsic data structure in union of vector subspaces, and thus are suffered from the curse of dimensionality.

In this chapter, novel methodologies which exploit linear dependencies among SPNs in their intrinsic vector subspaces, are presented. Such dependencies are encoded under sparse representations which are obtained by solving a LASSO problem with non-negativity constraint. We show that our proposed methodologies are particularly suited to image clustering by acquisition device, both in medium-scale and large-scale applications.

In Chapter 1, we draw the importance of image clustering by source camera in blind scenarios which means that no auxiliary information can be assumed except the image pixel values. Based on that only information, the forensic analyst is required to group images by their acquisition devices. Figure 3.1 draws a simple illustration on how a pool of images coming from two devices is clustered.

In the next section, the process of SPN extraction from image pixel values is described, which results in noise-like signals to be ready for clustering.

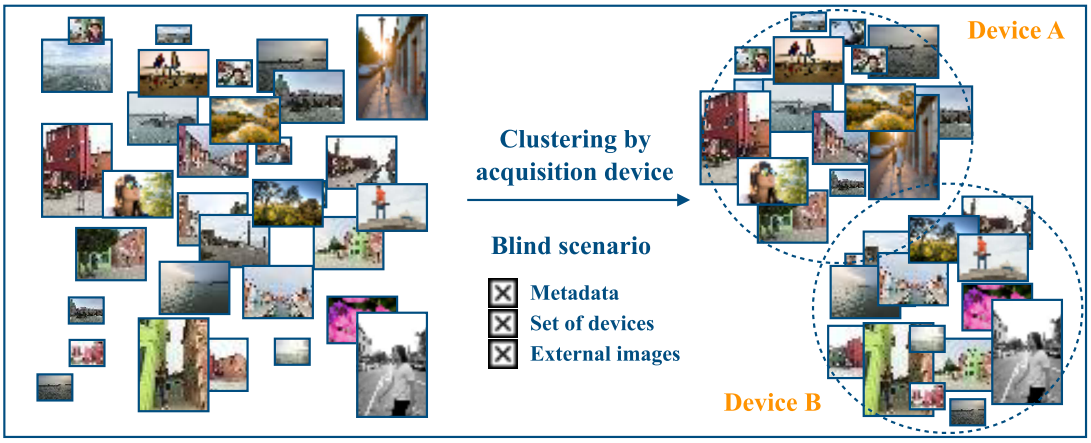


Figure 3.1: Image clustering by acquisition device in blind scenario.

3.1 Sensor Pattern Noise

A digital camera sensor is made of a discrete array of cells (pixels), which convert photons in output voltages. Voltages are then quantized into finite numerical values. Due to imperfections of pixels, the output image is overlaid by a pattern noise modulated by the scene light intensity [29]. Each digital image therefore intrinsically contains a SPN caused by sensor imperfections of the capturing device [28, 29].

Given a grayscale image $\mathbf{Y}$, its noise residual $\mathbf{W}$ can be extracted by a denoising filter. A simplified model of $\mathbf{W}$ can be expressed as follows [81, 29]:

$$\mathbf{W} = \mathbf{TYK} + \mathbf{\Xi}, \tag{3.1}$$

where $\mathbf{\Xi}$ is a the matrix of independently and identically distributed (i.i.d) Gaussian random variables, $\mathbf{T}$ is an attenuation matrix, and $\mathbf{K}$ is referred to as Photo-Response Non-Uniformity (PRNU) whose maximum likelihood estimate can be found in [29].

In theory, PRNU can be used to cluster images with respect to the acquisition device. However, in a blind scenario this is difficult due to two main problems. First, the Cramer-Rao Lower Bound on the variance of PRNU estimate indicates that a number of smooth and bright (but not saturated) images are required for each camera [29] and this condition is hardly satisfied. Indeed, the only available information is the noise residual $\mathbf{W}$ for each image, which contains not only the PRNU but also the additive noise $\mathbf{\Xi}$, which limits the reliability of traditional similarity measures used in conventional clustering algorithms. Several methods have been proposed for SPN enhancement [33, 34] but it has been confirmed by [2] that such methods are not suitable for unsupervised setting. Second, the dimension of camera fingerprints is usually high, due to the high resolution of camera sensors, thus their clustering requires huge computation and memory as long

as the number of data increases.

The first challenge can be addressed partially by using a good denoising filter to extract $\mathbf{W}$ such that the image details are greatly suppressed out. In this study, a noise residual $\mathbf{W}^c$, $c \in \{\text{red, green, blue}\}$ is extracted from each color channel of $\mathbf{Y}$, by exploiting the wavelet-based denoising filter commonly used in [28, 81, 29, 82], and then converted to one-channel noise residual $\mathbf{W}$:

$$\mathbf{W} = 0.3\mathbf{W}^{\text{red}} + 0.6\mathbf{W}^{\text{green}} + 0.1\mathbf{W}^{\text{blue}}.$$

As discussed in [81, 29], $\mathbf{W}$ contains non-unique artifacts possibly caused by demosaicing, JPEG blocky artifacts, row-wise and column-wise operation of sensors and processing circuits. The diagonal artifacts reported by [83] on Dresden dataset [3] can be also considered as non-unique, whose cause has not been comprehended yet. Non-unique artifacts make noise residuals of images of different cameras slightly correlated. They therefore should be suppressed to reduce false alarm rate. Following [81, 29], $\mathbf{W}$ is postprocessed by: row-wise and column-wise whitening, followed by Wiener filtering on its Fourier transform. After that, $\mathbf{W}$ is standardized as one-dimensional unit-norm signal and arranged as one column of $\mathbf{X} \in \mathbb{R}^{d \times n}$, n being the number of images, and d the number of selected pixels. Since the clustering algorithms typically work on $\mathbf{W}$, it is referred to as SPN throughout this chapter.

Existing approaches use normalized correlation to measure the similarity between two flattened SPNs $\mathbf{a}, \mathbf{b}$ of dimension d:

$$\rho(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^{d} (\mathbf{a}_i - \bar{\mathbf{a}}) \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)}{\sqrt{\sum_{i=1}^{d} (\mathbf{a}_i - \bar{\mathbf{a}})^2} \sqrt{\sum_{i=1}^{d} \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^2}}, \tag{3.2}$$

where scalars $\bar{\mathbf{a}}, \bar{\mathbf{b}}$ are the mean values of $\mathbf{a}$ and $\mathbf{b}$, respectively. Without loss of generality, if $\mathbf{a}, \mathbf{b}$ are normalized to have zero mean and unit norm, Equation (3.2) simply becomes:

$$\rho(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{d} \mathbf{a}_i \mathbf{b}_i,$$

which represents the cosine similarity between $\mathbf{a}$ and $\mathbf{b}$ in the pixel space.

The residual $\mathbf{W}$ is a noisy estimate of true camera fingerprint, thus the distributions of intra-class and inter-class correlations computed on $\mathbf{W}$ are heavily overlapped, making clustering algorithms less accurate. This challenge raises the need to eliminate inter-class data relationships and obtain unambiguous underlying data structure. The next section focuses on a powerful method which expresses each data point by a few linear relationships with other data points and extracts unambiguous representation which is afterwards proved to be effective for clustering SPNs.

3.2 Sparse Representation

Given a set of data points arranged into the columns of a matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, a data point $\mathbf{y}$ can be expressed as a linear combination of the columns of $\mathbf{X}$. A *sparse* combination reveals columns in the same subspace which $\mathbf{y}$ happens to lie into. Sparse Subspace Clustering (SSC) [84] finds a sparse representation of $\mathbf{y}$ by solving the following optimization problem:

$$\underset{\mathbf{z}}{\text{minimize}} \quad \|\mathbf{z}\|_1 \quad \text{subject to} \quad \mathbf{X}\mathbf{z} = \mathbf{y}. \tag{3.3}$$

If columns of $\mathbf{X}$ are contaminated by noise or not well distributed, $\mathbf{X}\mathbf{z} = \mathbf{y}$ might never be reached. Works in [85, 86] have shown that SSC can deal with noisy data if Equation (3.3) is reformulated as a LASSO problem:

$$\underset{\mathbf{z}}{\text{minimize}} \quad \|\mathbf{X}\mathbf{z} - \mathbf{y}\|_2^2 + \gamma\|\mathbf{z}\|_1, \tag{3.4}$$

where $\gamma \geq 0$ is a regularization hyperparameter.

Let us interpret sparse representation via a simple example. As illustrated in Figure 3.2, we have $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4] \in \mathbb{R}^{3 \times 4}$. We assume there are two subspaces spanned by $[\mathbf{X}_1, \mathbf{X}_2]$ and $[\mathbf{X}_3, \mathbf{X}_4]$. Without the regularization term, $\mathbf{y}$ can be expressed as linear combination of any 3 columns of $\mathbf{X}$. However, we would like to assign $\mathbf{y}$ to the closest subspace spanned by $[\mathbf{X}_1, \mathbf{X}_2]$, i.e., $\|\hat{\mathbf{y}} - \mathbf{y}\|_2 < \|\tilde{\mathbf{y}} - \mathbf{y}\|_2$, a preferable solution should satisfy:

$$\hat{\mathbf{y}} = \mathbf{z}_1\mathbf{X}_1 + \mathbf{z}_2\mathbf{X}_2.$$

Such solution can be reached if we encourage the sparseness of $\mathbf{z}$ by penalizing $\|\mathbf{z}\|_0$. Unfortunately, ℓ_0 optimization is usually intractable due to its non-convex and combinatorial nature. $\|\mathbf{z}\|_1$ can be used instead, being a good approximation of $\|\mathbf{z}\|_0$ [87]. Exploiting the ℓ_1 regularization term as in Equation (3.4) and properly selecting γ, SSC finds a sparse solution that minimizes the reconstruction error.

In the next section, we introduce a clustering method that leverages the power of sparse representations. This algorithm is particularly suited to medium-scale datasets.

3.3 Medium-Scale Clustering

3.3.1 The Proposed Optimization

SSC learns a sparse representation $\mathbf{z}$ of $\mathbf{y}$, whose non-zero entries indicate data points closest to the orthogonal projection of $\mathbf{y}$ onto the relevant subspace. We can *interpret* the magnitude of $\mathbf{z}_i$ as a similarity measure: the closer $\mathbf{X}_i$ is to $\hat{\mathbf{y}}$, the more it contributes to the reconstruction of $\mathbf{y}$, resulting in a larger value of $\mathbf{z}_i$. Back to the example in Figure

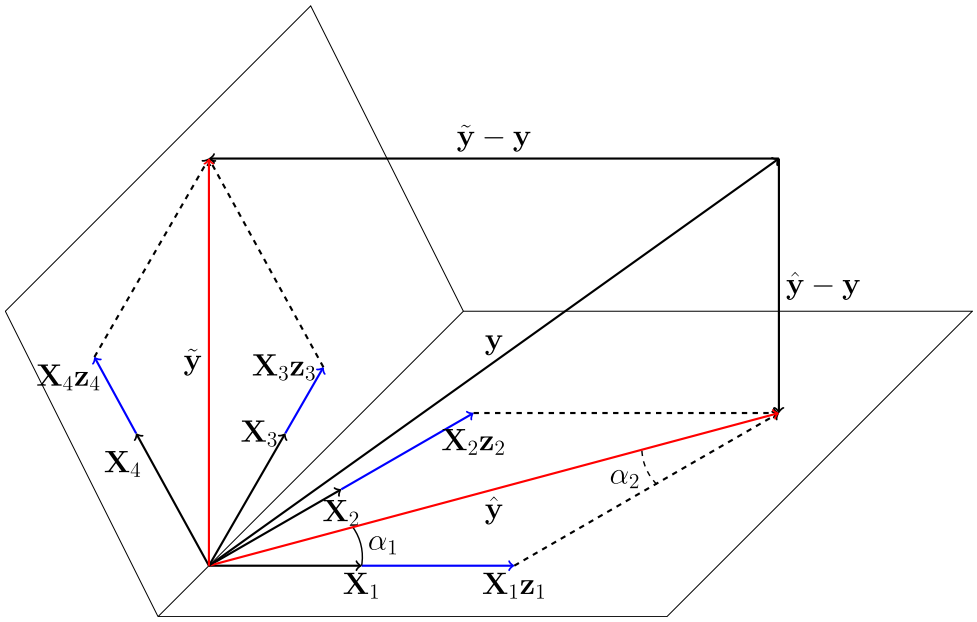


Figure 3.2: Geometric interpretation of solution of SSC.

3.2, denoted as $\alpha_i = \angle(\mathbf{X}_i, \hat{\mathbf{y}})$ the angle between $\hat{\mathbf{y}}$ and $\mathbf{X}_i$, it is easily to see that if $\alpha_i < \alpha_j$ then $|\mathbf{z}_i| > |\mathbf{z}_j|$:

$$\alpha_i < \alpha_j \Leftrightarrow \cos(\alpha_i) > \cos(\alpha_j) \Leftrightarrow \|\mathbf{z}_i\mathbf{X}_i\|_2 > \|\mathbf{z}_j\mathbf{X}_j\|_2$$

$$\Leftrightarrow |\mathbf{z}_i| > |\mathbf{z}_j| \,(\text{since } \|\mathbf{X}_i\|_2 = 1).$$

Thus, $|\mathbf{z}_i|$ is inversely proportional to α_i. The ℓ_1 regularization term encourages the sparseness of $\mathbf{z}$, whose non-zero entries should indicate data points *closest* to $\hat{\mathbf{y}}$. Due to the nature of ℓ_1 norm, however, negative and positive contributions are weighted equally. In Figure 3.3, the solution $\tilde{\mathbf{z}} = [-\mathbf{z}'_1, -\mathbf{z}'_2]^T$ might be chosen instead of $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2]^T$ as it is possible that $\|\tilde{\mathbf{z}}\|_1 < \|\mathbf{z}\|_1$. Note, in this example $\mathbf{z}_i > 0$ and $\mathbf{z}'_i > 0$, thus $\tilde{\mathbf{z}}$ contains negative entries. This is an unexpected solution since $\mathbf{X}'_1, \mathbf{X}'_2$ lie in another half-space, i.e., $\angle(\mathbf{X}'_1, \hat{\mathbf{y}}) > \pi/2$ and $\angle(\mathbf{X}'_2, \hat{\mathbf{y}}) > \pi/2$. In order to avoid this solution, all entries $\mathbf{z}_i$ can be constrained to non-negative values. Therefore, the optimal solution reveals data points lying in the subspace closest to $\mathbf{y}$ and positively correlated to the orthogonal projection of $\mathbf{y}$ onto that subspace. Another interesting property is that if $\mathbf{z}_i \geq 0$, $\mathbf{z}_j \geq 0$ and $\alpha_i < \alpha_j$ then $\mathbf{z}_i > \mathbf{z}_j$. Thanks to this observation, it is necessary to impose a non-negativity constraint on the optimization problem. By this particular difference to SSC, we refer to the proposed clustering method as Sparse Subspace Clustering with Non-negativity Constraint (SSC-NC).

For each column $\mathbf{X}_i$, a sparse representation $\mathbf{Z}_i$ is obtained such that $\mathbf{X}_i = \mathbf{X}\mathbf{Z}_i$.

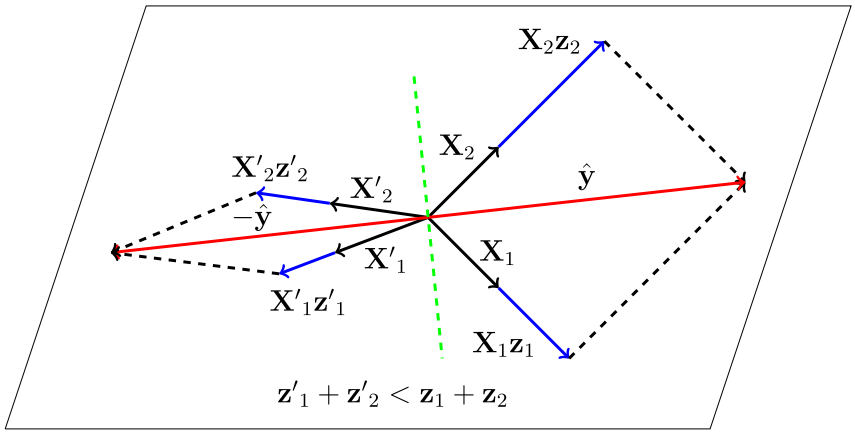


Figure 3.3: Geometric interpretation of negative solution of SSC.

To obtain a meaningful representation, a column should not be expressed by itself, thus requiring the constraint $\mathbf{Z}_{ii} = 0$. Accordingly, the following optimization problem is proposed in this study to extract sparse representation of $\mathbf{X}$:

$$\begin{aligned} \underset{\mathbf{Z}}{\text{minimize}} \quad & \frac{1}{2}\left\|\mathbf{XZ} - \mathbf{X}\right\|_F^2 + \gamma\left\|\mathbf{Z}\right\|_1 \\ \text{subject to} \quad & \text{diag}(\mathbf{Z}) = 0, \mathbf{Z} \geq 0, \end{aligned} \tag{3.5}$$

where $\gamma > 0$ is the regularization hyperparameter.

Many research efforts have been spent in solving the unconstrained version of Equation (3.5) [88]. The ℓ_1 minimization problem does not have an analytical solution; its solution instead has to be obtained numerically. Among the proposed algorithms, Augmented Lagrange Multiplier (ALM) generally converges faster under a wide range of data [88]. This study adopts Alternating Direction Method of Multipliers (ADMM) [89] to solve the problem in Equation (3.5), which couples the fast convergence of ALM with the decomposability property, which is fundamental for distributed implementation in large-scale problems. ADMM introduces a complementary variable $\mathbf{V}$ and re-formulates the unconstrained version of Equation (3.5) into the following equivalent form:

$$\begin{aligned} \underset{\mathbf{Z},\mathbf{V}}{\text{minimize}} \quad & \gamma\|\mathbf{V}\|_1 + \frac{1}{2}\|\mathbf{XZ} - \mathbf{X}\|_F^2 \\ \text{subject to} \quad & \mathbf{Z} = \mathbf{V}. \end{aligned} \tag{3.6}$$

Here, decomposability means that $\mathbf{Z}$ and $\mathbf{V}$ can be updated separately, possibly on a distributed system, thus constraints in Equation (3.5) can be imposed on $\mathbf{V}$. They are enforced during $\mathbf{V}$ update by Euclidean projections which are much simpler than ALM.

The augmented Lagrangian form of Equation (3.6) is

$$\mathcal{L}_\eta(\mathbf{Z}, \mathbf{V}, \mathbf{\Lambda}) = \gamma\|\mathbf{V}\|_1 + \frac{1}{2}\|\mathbf{XZ} - \mathbf{X}\|_F^2 + \langle \mathbf{\Lambda}, \mathbf{Z} - \mathbf{V}\rangle + \frac{\eta}{2}\|\mathbf{Z} - \mathbf{V}\|_F^2.$$

where $\mathbf{\Lambda} \in \mathbb{R}^{n\times n}$ is the Lagrangian multiplier and $\eta > 0$ is the augmented Lagrangian hyperparameter. ADMM iteratively optimizes $\mathbf{Z}, \mathbf{V}$ in an alternate fashion, by keeping one variable fixed and updating the others:

$$\begin{aligned}
\mathbf{Z}^{t+1} &= \arg\min_{\mathbf{Z}} \mathcal{L}_\eta\left(\mathbf{Z}, \mathbf{V}^t, \mathbf{\Lambda}^t\right), \\
\mathbf{V}^{t+1} &= \arg\min_{\mathbf{V}} \mathcal{L}_\eta\left(\mathbf{Z}^{t+1}, \mathbf{V}, \mathbf{\Lambda}^t\right), \\
\mathbf{\Lambda}^{t+1} &= \mathbf{\Lambda}^t + \eta\left(\mathbf{Z}^{t+1} - \mathbf{V}^{t+1}\right).
\end{aligned}$$

It is straightforward to demonstrate that $\mathbf{Z}$ can be updated by solving the linear equation:

$$(\mathbf{X}^T\mathbf{X} + \eta\mathbf{I})\mathbf{Z} \;=\; (\mathbf{X}^T\mathbf{X} - \mathbf{\Lambda} + \eta\mathbf{V}),$$

using Cholesky decomposition of $\mathbf{X}^T\mathbf{X} + \eta\mathbf{I}$. On the other hand, solution of $\mathbf{V}$ at each iteration is obtained through soft thresholding operator S defined as:

$$S_\nu(a) \;=\; \begin{cases} a - \nu & a > \nu \\ a + \nu & a < -\nu \\ 0 & |a| \le \nu \end{cases}.$$

Details of its update are provided in Appendix 7.2. After $\mathbf{V}$ update, the two following operators are applied to project $\mathbf{V}$ into the feasible set of solutions:

$$\Pi_D(\mathbf{M}_{ij}) \;=\; \begin{cases} \mathbf{M}_{ij} & i \neq j \\ 0 & i = j, \end{cases} \tag{3.7}$$

$$\Pi_N(\mathbf{M}_{ij}) \;=\; \begin{cases} \mathbf{M}_{ij} & \mathbf{M}_{ij} \ge 0 \\ 0 & \mathbf{M}_{ij} < 0. \end{cases} \tag{3.8}$$

The optimization procedure is reported in Algorithm 1: it converges efficiently to an acceptable solution as $\|\mathbf{Z} - \mathbf{V}\|_\infty \to 0$.

To visually compare sparse representation and normalized correlation matrix, we conduct an analysis on synthetic noise and another one on realistic noise. Synthetic noise is extracted from images generated by the simple imaging model described in [29] for smooth images (without the attenuation factor $\mathbf{T}$), that is $\mathbf{Y} = \mathbf{Y}^{(0)} + \mathbf{Y}^{(0)}\mathbf{K} + \mathbf{\Xi}$. The clean image $\mathbf{Y}^{(0)}$ is uniform, having pixel value of 0.9 (relatively bright). $\mathbf{K}_{ij}$ and $\mathbf{\Xi}_{ij}$ are reasonably assumed as white Gaussian noise. As the signal $\mathbf{K}$ is generally weaker than

Algorithm 1 Constrained LASSO

procedure CONSTRAINED_LASSO($\mathbf{X}, \gamma, \eta$)

initialize: $\mathbf{Z} \leftarrow 0, \mathbf{V} \leftarrow 0, \mathbf{\Lambda} \leftarrow 0, \varepsilon \leftarrow 10^{-4}$

while convergence condition is not satisfied **do**

Fix the others, update $\mathbf{Z}$

$$\mathbf{Z} \leftarrow (\mathbf{X}^T\mathbf{X} + \eta\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X} - \mathbf{\Lambda} + \eta\mathbf{V})$$

Fix the others, update $\mathbf{V}$

$$\mathbf{V}_{ij} \leftarrow S_{\frac{\gamma}{\eta}}\left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right)$$

$$\mathbf{V}_{ij} \leftarrow \Pi_D(\Pi_N(\mathbf{V}_{ij}))$$

Fix the others, update $\mathbf{\Lambda}$: $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \eta\,(\mathbf{Z} - \mathbf{V})$

Check convergence condition: $\|\mathbf{Z} - \mathbf{V}\|_\infty < \varepsilon$

end while

return Z

end procedure

$\mathbf{\Xi}$, the variance of $\mathbf{K}_{ij}$ is selected as 0.001 and the variance of $\mathbf{\Xi}_{ij}$ is 0.1 (for pixel values in $[0, 1]$). We simulate the situation of 5 cameras corresponding to 5 different $\mathbf{K}$ patterns, considering 100 images for each camera, thus resulting into 500 different $\mathbf{\Xi}$ patterns. After that, we apply the same wavelet-based denoising filter to extract synthetic noise. A sample of extracted synthetic noise is depicted in Figure 3.4 (a). For the realistic setting, we select 5 cameras from the Vision dataset [90], 100 images for each camera, and apply the same denoising procedure. In Figure 3.4 (b) an example of realistic noise is shown. We intentionally group noise residuals of the same camera so that the representation matrix is easily observable. We show sparse representation matrix of synthetic noise in Figure 3.4 (c), and of realistic noise in Figure 3.4 (e). The dense representation matrix in Figure 3.4 (d) and 3.4 (f) are obtained by computing pair-wise normalized correlation for synthetic noise and for realistic noise, respectively. Noticeably, solving the problem in Equation (3.5) obtains meaningful representation where inter-class relations are effectively removed, revealing clearer block-diagonal structure compared to normalized correlation.

3.3.2 Clustering

It is worth to emphasize again that in blind scenario no auxiliary information including the number of acquisition devices, i.e. number of clusters, is given. In this study, the

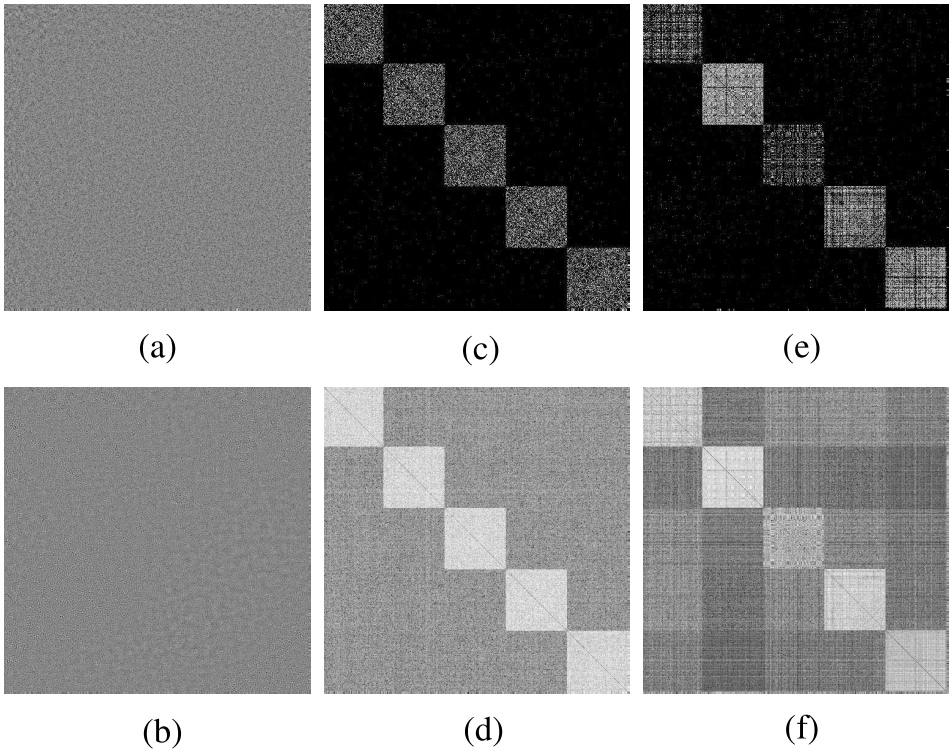


Figure 3.4: Visual comparison of sparse representation and dense representation (obtained by normalized correlation): (a) synthetic noise sample, (b) realistic noise sample, (c) sparse representation matrix of synthetic noise, (d) dense representation matrix of synthetic noise, (e) sparse representation matrix of realistic noise, (f) dense representation matrix of realistic noise.

number of clusters is found automatically based on the sparse representation matrix $\mathbf{Z}$.

The sparse representation matrix captures asymmetric relationships among data points, i.e., $\mathbf{Z}_{ij} \neq \mathbf{Z}_{ji}$. For our clustering purpose, we build a weighted undirected graph $\mathbf{G}$ from $\mathbf{Z}$ as $\mathbf{G} = (\mathbf{Z} + \mathbf{Z}^T)/2$. $\mathbf{G}_{ij} \neq 0$ implies a linear dependency between node i and node j, while $\mathbf{G}_{ij} = 0$ implies a linear independence. To obtain the final segmentation, the spectral clustering described in [91] is applied to partition $\mathbf{G}$ into κ connected components or clusters. In ideal cases, the number of connected components in $\mathbf{G}$ is indeed the number of zero eigenvalues of the normalized graph Laplacian $\mathbf{L}$ (Proposition 4 in [92]), where

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{G}\mathbf{D}^{-1/2},$$

and $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{G}_{ij}$, i.e., the degree of i-th vertex.

κ can be inferred from the number of small eigenvalues. Therefore, we adopt an approach based on *eigengap heuristic* [92] to infer the number of clusters. The idea is to choose κ such that sorted eigenvalues $\lambda_2, \ldots, \lambda_\kappa$ (λ_1 is always 0) of $\mathbf{L}$ are relatively small

but $\lambda_{\kappa+1}$ is large, creating a biggest gap among consecutive eigenvalues. More classical algorithms exist for κ estimation, such as finding feasible segmentations with multiple values of κ and select the best segmentation based on some criterion, such as Silhouette coefficients [93] or gap statistics [94]. Nevertheless, we found that our approach is more efficient since we find only one segmentation. It is worth noting that eigendecomposition is computed only once for both κ estimation and spectral clustering. Moreover, the estimate of κ is more accurate, because the sparseness of $\mathbf{G}$ can be controlled by the regularization hyperparameter γ. This hyperparameter controls the natural trade-off between false and true discovery rates. Larger γ results in sparser solutions, making few false discoveries. At the same time, we expect to have many true discoveries by making the solution less sparse.

We define a criterion based on *normalized cut* (NCut) and *eigengap*. NCut is originally introduced in [95] for image segmentation problem, and is defined as:

$$\mathrm{NCut}_\kappa \quad = \quad \frac{1}{\kappa}\sum_{c=1}^{\kappa}\frac{\mathbf{G}(A_c,\bar{A}_c)}{\mathbf{G}(A_c,\bar{A}_c)+\mathbf{G}(A_c,A_c)},$$

where $\kappa, \mathbf{G}$ refer to the number of clusters and to the affinity matrix, respectively. A_c is the set of indices of SPNs belonging to cluster c. Note that $\bar{A}$ is the complement of A and $\mathbf{G}(A,B) = \sum_{i\in A, j\in B}\mathbf{G}_{ij}$. Minimizing NCut is indeed equivalent to mimimizing inter-connections and maximizing intra-connections. The parameter γ should be selected so that it maximizes the eigengap $\Gamma_\kappa = \lambda_{\kappa+1} - \lambda_\kappa$ and minimizes neighboring eigengaps $\Gamma_{\kappa-1}, \Gamma_{\kappa+1}$. By composing two defined criteria, NCut and eigengap, we seek γ minimizing the following cost function:

$$J(\kappa) \quad = \quad \mathrm{NCut}_\kappa + \frac{1}{\Gamma_\kappa} + \Gamma_{\kappa+1} + \Gamma_{\kappa-1}. \tag{3.9}$$

As NCut_κ ranges in $[0,1]$, we linearly scale λ_i to $[0,1]$ so that no weighting is needed for terms in Equation (3.9). γ can be found by brute-force search over T discrete values of γ in $[\gamma_{\min}, \gamma_{\max}]$. It is well-known that solutions of Lasso are piece-wise linear with respect to γ [96]. This property also holds for Constrained_Lasso, where solutions are constrained to be zero-diagonal and non-negative. We exploit the linearity of solution path and design a warm start Constrained_Lasso. In particular, the solution at γ_{t-1} $(1 < t \leq T)$ is used to initialize the solver at γ_t, and Cholesky decomposition is computed only once. This computational saving is extremely important on large-scale sets.

In summary, the proposed algorithm SSC-NC learns sparse representations of noise residuals. This approach provides a good instrument to discover structures on high-dimensional data, but shows a major drawback on scalability, since all data must be loaded on RAM. In Section 3.5 the computational complexity of Algorithm 1 will be

demonstrated to be in the order of n^3, n being the number of fingerprints. Empirically, this is acceptable only for datasets with $n \leq 6000$. In the following section, we extend SSC-NC to large-scale contexts.

3.4 Large-Scale Clustering

The problem scaling to large datasets can be addressed in different ways. A possibility is applying classical methods such as DBSCAN [97], BIRCH [98], or CURE [99]. Unfortunately, these methods make use of similarity measures such as Euclidean distance or cosine similarity, which can be computed accurately and efficiently on small data batches. SSC instead, requires to learn data relationship from the entire dataset. One of the first contributions to scalable SSC is [100], later extended in [101]. In these works, the underlying idea is that a small number of data samples are sufficient to learn sparse representations and forming base clusters. The remaining data can be later associated to the existing clusters using sparse coding. However, there is no guarantee that a subset of data allows discovering all clusters. More recently, in the work Orthogonal Matching Pursuit (OMP) [102] the authors proposed to replace the complex ℓ_1 norm optimization, but their method guarantees the optimality only if the subspaces are independent; moreover, it does not solve the memory problem.

In this section, a new methodology called large-scale SSC (LS-SSC) is introduced to cluster images in large-scale contexts. First, we address the memory issue using a *divide-and-conquer* strategy, so that compact clusters could be discovered on small data batches. Since SSC learns the linear dependencies among data points, insufficient data might result in inappropriate relationships. This problem can be solved by preserving only confident data dependencies and discarding outliers. Obviously, some clusters might be hidden and undiscovered even after visiting all data batches. We therefore re-cycle previously discarded data in order to eventually discover all clusters. This process is followed by *merging* and *attraction* phases which finalize clustering results. The overall large-scale clustering includes three main phases, which are depicted in Figure 3.5.

Splitting, Clustering and Recycling

The baseline strategy of our large-scale clustering is the divide-and-conquer paradigm, which breaks an intractable problem into several smaller tractable problems. We randomly split the set of all SPNs $\mathcal{X}$ into B batches of equal size, $\mathcal{X} = \left\{\mathcal{X}^l\right\}_{l=1,\dots,B}$, where B is originally set to $\lceil \frac{n}{p} \rceil$ and p is the batch size. Only one data batch at a time is loaded on RAM. We then apply Algorithm 1 on the data batch to learn sparse representations among SPNs.

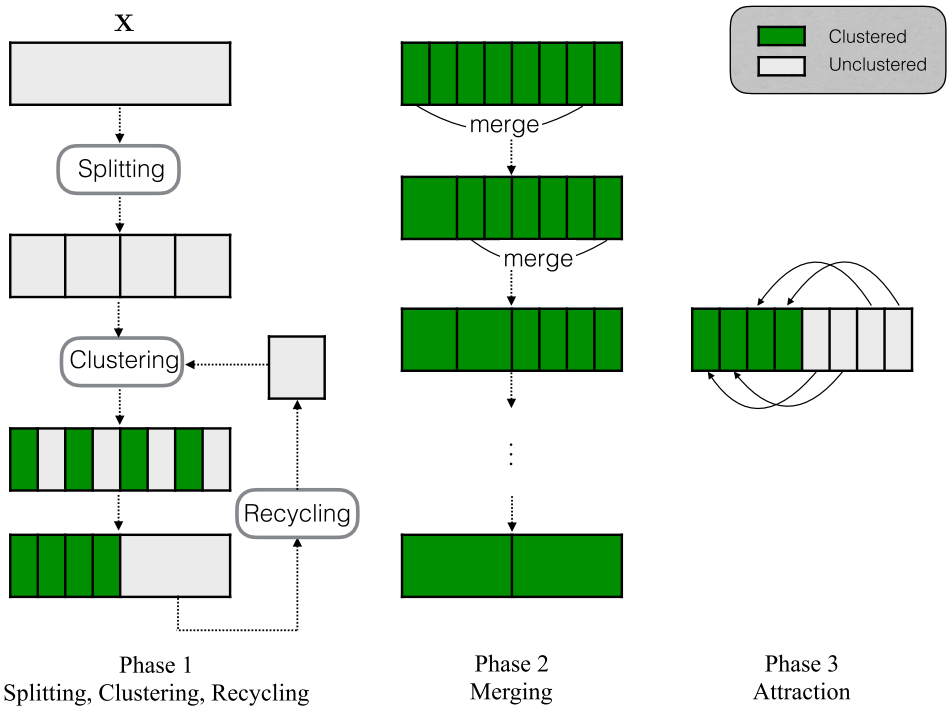


Figure 3.5: Schema of the proposed LS-SSC.

We hereby refer to *cluster purity* as the quality of a cluster. A pure cluster should contain only SPNs of the same camera. The main purpose of this phase is to extract small-size but pure clusters that can be later merged to form larger clusters. As a result of splitting, a SPN might not be well reconstructed by only SPNs from the same camera. To minimize the reconstruction error, the algorithm might select SPNs from multiple cameras. Such representations are considered as *outliers*, similarly to [103].

Let us now consider a sparse representation matrix as a directed graph: outliers have connections to both outliers and inliers, while inliers have connections to inliers only. If we perform a random walk on the graph, the probability of ending at inliers is therefore higher than ending at outliers. We apply the random walk algorithm described in [103] with 1000 steps to acquire the state probabilities, we model such probabilities as a normal distribution, and we keep 80% of the distribution as inliers, thus classifying the rest as outliers.

To guarantee the purity of clusters, we avoid spectral clustering, but we attempt to localize dense regions using the interpretability property of our sparse representation. Accordingly, large values indicate *closest* SPNs. Since our target is to discover small-size but pure clusters, we can further simplify the graph by retaining only K largest entries on each column of the sparse representation matrix, and setting other entries to zero.

Each remaining SPN is located in a region of K nearest neighbors, and SPNs in the same cluster should have common neighbors, forming a dense region.

After that, we apply DBSCAN [97] to discover dense regions. This classical clustering technique is computationally feasible for large-scale datasets and does not require the number of clusters to be known. Two parameters need to be indicated as input of DBSCAN: radius ϵ and minimum number of neighbors $MinPts$. The radius ϵ should be selected on the basis of K largest values on each column of the sparse representation matrix, while $MinPts$ must be smaller or equal to K. If ϵ is too small, this results in many clusters. Conversely, very limited number of clusters are discovered, complicating the recycling process. On the other hand, if $MinPts > K$, there is no cluster discovered by DBSCAN. We empirically found that setting $MinPts = K$ and ϵ equal to the mean of non-zero entries, allows discovering pure clusters.

After clustering, we obtain the set of inliers and outliers, where inliers are used for the merging phase and outliers are fed to the recycling process. The aim of recycling is to combine outliers from each batch and feed them back to the clustering process, thus increasing the chance to discover hidden clusters. For that reason, clustering and recycling can be seen as an iterative procedure, as outlined in Algorithm 2.

Merging

In the first phase, by increasing K we obtain larger clusters at the expense of a lower cluster purity. Conversely, we yield small-size pure clusters. The latter is preferable, as small-size clusters (subclusters) can be merged efficiently to form larger subclusters.

Let $\mathbf{W}^A$ and $\mathbf{W}^B$ be two noisy SPNs of dimension d, i.e., two singleton subclusters, reasonably assumed to follow a normal distribution since the denoising filter extracts stationary Gaussian noise in wavelet domain (see Appendix A of [28]). $\mathbf{K}^A$ and $\mathbf{K}^B$ are the noise-free SPNs residing in $\mathbf{W}^A$, $\mathbf{W}^B$. The merging problem can be formulated as a classical hypothesis test:

$$\begin{aligned} \mathrm{H}_0 &: \mathbf{K}^A \neq \mathbf{K}^B, \\ \mathrm{H}_1 &: \mathbf{K}^A = \mathbf{K}^B = \mathbf{K}. \end{aligned}$$

Under null hypothesis, $\rho\left(\mathbf{W}^A, \mathbf{W}^B\right) \sim \mathcal{N}(0, \frac{1}{d})$ according to the Central Limit Theorem (CLT). Two subsclusters can be merged if their normalized correlation exceeds $\frac{1}{\sqrt{d}} Q^{-1}(P_{FA})$, where $Q(t)$ is the probability that a standard normal variable is larger than t and P_{FA} is the expected false alarm rate [82]. More generally, if each cluster contains more than one SPN, $\mathbf{W}^A$ and $\mathbf{W}^B$ represent respectively the subcluster centroids. Under alternative hypothesis, the correlation between $\mathbf{W}^A$ and $\mathbf{W}^B$ increases if the cardinality of each subcluster increases, as random noise is effectively suppressed by averaging. Obvi-

Algorithm 2 Splitting, clustering, recycling

procedure SPLITTING_CLUSTERING_RECYCLING
input: $\mathcal{X}, p, R, K, \gamma, \eta$ ▷ $\mathcal{X}$: dataset, p: batch size, R: number of recycling steps, K: number of nearest neighbors
output: $\mathcal{X}_{\text{in}}$, $\mathcal{X}_{\text{out}}$ ▷ set of clustered and unclustered SPNs
$\mathcal{X}_{\text{in}} \leftarrow \emptyset$
$B \leftarrow \lceil \frac{n}{p} \rceil$
Split $\mathcal{X}$ into $\left\{\mathcal{X}^l\right\}_{l=1,\ldots,B}$
$\mathcal{X}^l_{\text{out}} \leftarrow \emptyset$, ▷ $l = 1, \ldots, B$
for $l = 1 \rightarrow B$ **do**
$\tilde{\mathcal{X}}^l_{\text{in}}, \tilde{\mathcal{X}}^l_{\text{out}} \leftarrow$ PARTITION$(\mathcal{X}^l, K, \gamma, \eta)$
Append $\tilde{\mathcal{X}}^l_{\text{in}}$ to $\mathcal{X}_{\text{in}}$ and append $\tilde{\mathcal{X}}^l_{\text{out}}$ to $\mathcal{X}^l_{\text{out}}$
end for
$t \leftarrow B$, $\tilde{B} \leftarrow B$
repeat
$\mathcal{X}^t_{\text{out}} \leftarrow \emptyset$, $\mathcal{X}^t \leftarrow \emptyset$
for $l = 1 \rightarrow \tilde{B}$ **do**
Pop out randomly $s^l = \frac{|\mathcal{X}^l_{\text{out}}| \times p}{\sum_{i=1}^{\tilde{B}} |\mathcal{X}^i_{\text{out}}|}$ SPNs from $\mathcal{X}^l_{\text{out}}$
Append s^l SPNs to $\mathcal{X}^t$
end for
$\tilde{\mathcal{X}}^t_{\text{in}}, \tilde{\mathcal{X}}^t_{\text{out}} \leftarrow$ PARTITION$(\mathcal{X}^t, K, \gamma, \eta)$
Append $\tilde{\mathcal{X}}^t_{\text{in}}$ to $\mathcal{X}_{\text{in}}$ and append $\tilde{\mathcal{X}}^t_{\text{out}}$ to $\mathcal{X}^t_{\text{out}}$
$t \leftarrow t + 1$, $\tilde{B} \leftarrow \tilde{B} + 1$
until $t \geq B + R$
$\mathcal{X}_{\text{out}} \leftarrow \left\{\mathcal{X}^l_{\text{out}}\right\}_{l=1,\ldots,B+R}$
end procedure
procedure PARTITION
input: $\mathcal{X}, K, \gamma, \eta$ ▷ $\mathcal{X}$: dataset, K: number of nearest neighbors
output: $\mathcal{X}_{\text{in}}$, $\mathcal{X}_{\text{out}}$ ▷ set of clustered and unclustered SPNs
Load SPNs in $\mathcal{X}$ to $\mathbf{X}$ ▷ $\mathbf{X}$: matrix of SPNs
$\mathbf{Z} \leftarrow$ CONSTRAINED_LASSO$(\mathbf{X}, \gamma, \eta)$
Remove outliers, obtain $\tilde{\mathbf{Z}}$. Append outliers to $\mathcal{X}_{\text{out}}$
Keep only K largest entries on each column of $\tilde{\mathbf{Z}}$, obtain $\tilde{\mathbf{Z}}_{\text{KNN}}$
Apply DBSCAN to discover clusters
Append inliers to $\mathcal{X}_{\text{in}}$
Append outliers to $\mathcal{X}_{\text{out}}$
end procedure

ously, the merging phase will be more reliable if one knows not only the null distribution but also the alternative distribution.

To determine the alternative distribution, [104, 2] established a parametric model with some statistical assumptions and determined model parameters. For instance, the true SPNs are assumed to be additive noise [104, 2], and the WGN of a camera presents always the same variance [104]. Nevertheless, if those assumptions are not guaranteed, and usually they are not, parameter estimation becomes extremely difficult.

We resort the merging problem into finding a threshold value τ that is able to exclude the null hypothesis and to adapt to the variation of the alternative hypothesis, based on real data. This is achieved by taking into account the cardinality and intra-class correlation within each subcluster. Let $\mathbf{X}^A$ and $\mathbf{X}^B$ be the two matrices containing n_A and n_B SPNs of each subcluster, and ρ_A and ρ_B be the intra-class correlation within these subclusters. We learn the threshold adaptiveness via linear regression:

$$\mathcal{R}\left(n_A, n_B, \rho_A, \rho_B\right) \;=\; \left[n_A, n_B, \rho_A, \rho_B\right]\mathbf{w} + b,$$

where $\mathbf{w} \in \mathbb{R}^{4\times 1}$ and $b \in \mathbb{R}$ are weights and bias, respectively. From real data, we calculate ρ_A, ρ_B and the regression output $\mathcal{R}(\cdot)$ as follows:

$$\begin{aligned}
\rho_A &= \frac{1}{n_A(n_A-1)}\sum_{i=1}^{n_A}\sum_{j=1,j\neq i}^{n_A}\rho(\mathbf{X}_i^A, \mathbf{X}_j^A),\\
\rho_B &= \frac{1}{n_B(n_B-1)}\sum_{i=1}^{n_B}\sum_{j=1,j\neq i}^{n_B}\rho(\mathbf{X}_i^B, \mathbf{X}_j^B),\\
\mathcal{R}(\cdot) &= \frac{\rho(\bar{\mathbf{X}}^A, \bar{\mathbf{X}}^B)}{2},
\end{aligned}$$

where $\bar{\mathbf{X}}^A, \bar{\mathbf{X}}^B$ are respectively two subcluster centroids, i.e., mean of columns in $\mathbf{X}^A$ and $\mathbf{X}^B$. The estimate of $\mathcal{R}(\cdot)$ is interpreted as the central value between mean of null and alternative distribution. The final regressor learnt from real data (we will mention this development set in Section 3.6.1) has the form:

$$\mathcal{R}\left(n_A, n_B, \rho_A, \rho_B\right) \;=\; 0.0016\, n_A + 0.0016\, n_B + 2.2474\, \rho_A + 2.2474\, \rho_B - 0.0474.$$

Careful readers will notice that the regressor is symmetric in terms of cluster role, i.e., $\mathcal{R}\left(\, n_A, n_B, \rho_A, \rho_B \,\right) = \mathcal{R}\left(\, n_B, n_A, \rho_B, \rho_A \,\right)$. This is achieved by augmenting the training data with the role of two clusters exchanged. The threshold τ is finally calculated as:

$$\tau \;=\; \max\left\{\frac{1}{\sqrt{d}}Q^{-1}(P_{FA}), \mathcal{R}\left(\cdot\right)\right\},$$

where P_{FA} is chosen as 0.001 (0.1% false alarm rate).

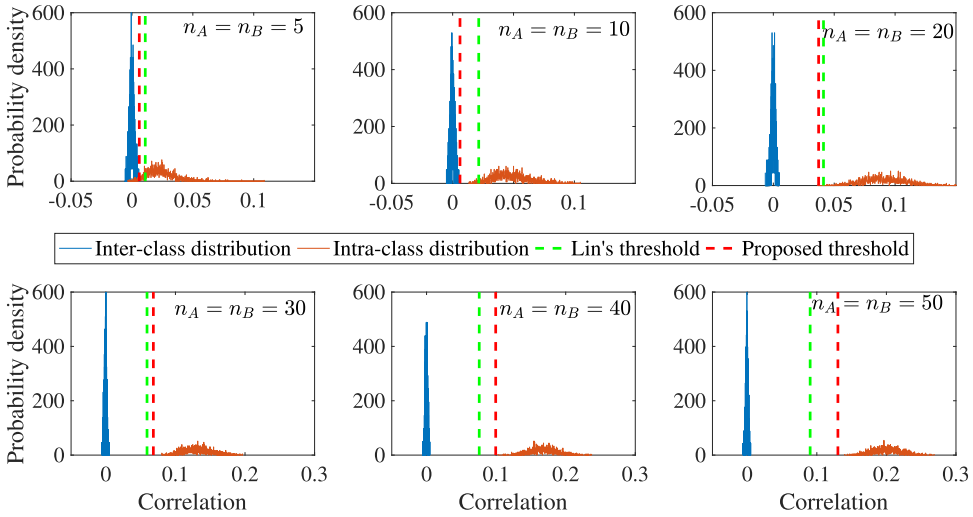


Figure 3.6: Comparison of the proposed threshold and Lin's threshold [2] under two cameras: Kodak M1063 and Nikon CoolPix S710. Better viewed in color.

The merging phase is conducted as an iterative procedure, which respectively selects pairs of subclusters having maximum centroid correlation and compares them to τ. If the correlation is larger than τ, the two subclusters are merged and relevant information is updated. The algorithm stops when no more pairs of subclusters exist that satisfy the merging condition.

In practical cases, a good regressor might not be linear, but for n_A, n_B within a reasonably small range, τ can be fitted by a linear function. Therefore, to calculate a reliable τ, we set $n_A = \min\{\tilde{n}_A, 50\}$ and $n_B = \min\{\tilde{n}_B, 50\}$ where $\tilde{n}_A, \tilde{n}_B$ are actual cluster cardinalities, and calculate ρ_A, ρ_B using bounded sets of SPNs. The quantity 50 is suggested as a minimal cardinality for SPN estimation [105, 28].

To demonstrate the effectiveness of the proposed threshold, we compare with Lin's threshold [2], which was previously shown to be superior than thresholds in [37] and [106]. We select two cameras, namely Kodak M1063 and Nikon CoolPix S710 from Dresden database [3]. We randomly split images of one camera into two parts to simulate two same-camera subclusters. Images from different cameras are used to create cross-camera subclusters. The process is replicated 2000 times and intra-class and inter-class correlations are collected. Figure 3.6 shows how the proposed threshold and the threshold in [2] (Lin's threshold) separate null and alternative distribution. When the cardinality of same-camera subclusters increases, the alternative distribution shifts towards the right, while the null distribution is centered at 0. The proposed threshold consistently splits the two distributions, while Lin's threshold tends to be unnecessarily confident when two distributions are close. An interesting behavior of the proposed threshold and Lin's threshold

is their adaptiveness to distribution shifting.

Attraction

In attraction phase, we assign remaining SPNs to available clusters. Let us denote $\mathbf{C} = \left[\bar{\mathbf{C}}_1, \bar{\mathbf{C}}_2, \ldots, \bar{\mathbf{C}}_L\right] \in \mathbb{R}^{d \times L}$ the matrix containing centroids of L final clusters, and $\mathbf{X}^{\text{out}} \in \mathbb{R}^{d \times U}$ the data matrix containing U unclustered SPNs. Since the quality of camera SPNs is generally non-homogeneous, cluster assignment should be performed for high-quality SPNs first, in order to minimize assignment errors. The cluster membership $l, 1 \leq l \leq L$ of SPN $\mathbf{X}_i^{\text{out}}$, $1 \leq i \leq U$ is obtained iteratively by finding at each step the pair l and i such that $\rho(\mathbf{X}_i^{\text{out}}, \bar{\mathbf{C}}_l)$ is maximum and greater than $Q^{-1}(P_{FA})$ which is the threshold used to exclude null hypothesis in merging phase. After being attracted $\mathbf{X}_i^{\text{out}}$ is discarded, otherwise it is labeled to as *unclustered*. Since the remaining SPNs to be merged have been classified as outliers after recycling, we can expect that they are low-quality samples. Therefore, to reduce false alarm rate, the cluster centroid is updated only when its cardinality does not exceed 50, consistently with the empirical value used in the merging phase.

Eventually, we obtain the cluster memberships of camera SPNs in a large-scale database and a number of unclustered SPNs.

3.5 Computational Complexity

This section discusses on the time complexity of our proposed SSC-NC, LS-SSC and two recent works: correlation clustering with consensus (CCC) [46] and Lin's large-scale method (Lin-LS) [2].

SSC-NC. SSC-NC is composed by Constrained_Lasso and spectral clustering. Constrained_Lasso consists of Cholesky decomposition, linear equation solving and soft thresholding. In the worst case, Cholesky decomposition requires $n^3/3$ flops. Solving linear equations requires $2n^2$ flops of forward and backward substitutions. Soft-thresholding operation on n^2 variables requires n^2 computations. Let T_1 be the bound number of iterations, total cost of Constrained_Lasso is $\mathcal{O}\left(n^3/3 + 3T_1 n^2\right)$. Spectral clustering consists of maximum $\mathcal{O}\left(n^3\right)$ computations for eigendecomposition and $\mathcal{O}\left(T_2 \kappa^2 n\right)$ for K-means clustering on n κ-dimensional eigenvectors, where T_2 is the bound number of iterations in K-means and κ is the number of clusters. The time complexity of SSC-NC is $\mathcal{O}\left(4n^3/3 + 3T_1 n^2 + T_2 \kappa^2 n\right)$.

CCC. Similarly to typical clustering methods, CCC computes the correlation matrix which costs $\mathcal{O}\left(n^2\right)$. Correlation clustering are afterwards carried out by Adaptive Label Iterated Conditional Modes (AL-ICM) [107]. AL-ICM, a greedy algorithm, operates in it-

erative mode. Every SPN is initially assigned to a unique label. At each iteration, AL-ICM assigns to a SPN the label of its closest SPNs. This process is repeated until convergence where no label is updated. If T_3 is the bound number of iterations, the time complexity of AL-ICM is bounded to $\mathcal{O}(T_3 n^2)$. In CCC, correlation clustering is performed Q times where Q is the number of similarity thresholding values. Multiple base clusterings are combined to find the final clustering agreement by Weighted Evidence Accumulation Clustering (WEAC) [108]. The time complexity of WEAC is $\mathcal{O}((Q + \log n)n^2 + Qn)$. Finally, m obtained clusters are refined via a merging step which costs $\mathcal{O}(m^2 \log m)$. Total cost of CCC is $\mathcal{O}((QT_3 + Q + \log n + 1)n^2 + Qn + m^2 \log m)$.

LS-SSC. In large-scale contexts, we suppose that RAM can cache only p SPNs. The dataset is split into B batches, $B = \lceil \frac{n}{p} \rceil$. Clustering each batch requires running Constrained_Lasso, finding K nearest neighbors and DBSCAN. Finding K nearest neighbors requires sorting each column of sparse representation matrix, which is $\mathcal{O}(p^2 \log p)$. In the worst case, DBSCAN visits p points and scans for their neighbors, which costs $\mathcal{O}(p^2)$. Total cost of clustering B batches is $\mathcal{O}(B[p^3/3 + (3T_1 + \log p + 1)p^2])$. In our large-scale experiments, recycling step is replicated $B/2$ times on batches of size p. Merging and attracting phase work similarly to agglomerative hierarchical clustering, and their time complexity is respectively $\mathcal{O}(L^2 \log L)$ and $\mathcal{O}(UL \log U)$, where L is the number of discovered clusters after the first phase and U is the number of unclustered images. Total cost of LS-SSC is $\mathcal{O}(\ 1.5B^2[p^3/3 + (3T_1 + \log p + 1)p^2]\ + L^2 \log L + UL \log U\ )$.

Lin-LS. The time complexity of Lin-LS is analyzed for the first iteration. In the coarse step, the correlation calculation of B batches requires $\mathcal{O}(Bp^2)$. If the correlation matrix $n \times n$ has E non-zero entries, Graclus partitioning algorithm [109] has the time complexity of $\mathcal{O}(pE/n)$. Since the number of clusters in coarse step is fixed to $n^{1/4}$, the calculation of correlation matrix in the fining step costs $\mathcal{O}\left(n^{1/4}b^2\right)$ where b is the average size of clusters. Markov Clustering Algorithm (MCL) applied on $n^{1/4}$ coarse clusters is bounded to $\mathcal{O}\left(n^{1/4}bK^2\right)$, where K, for abuse of notation, is the maximal number of nonzero entries on each column of the binarized correlation matrix. Similarly to LS-SSC, merging and attraction of Lin-LS can be approximated to $\mathcal{O}(L^2 \log L)$ and $\mathcal{O}(UL \log U)$ where L is the discovered number of clusters and U refers to the number of unclustered SPNs. Since both LS-SSC and Lin-LS aim to obtain high-quality clusters of small size, we can equalize U, L in LS-SSC and U, L in Lin-LS for easy comparison. The first iteration of Lin-LS totally costs $\mathcal{O}\left(Bp^2 + (K^2 b + b^2)n^{1/4} + pE/n + L^2 \log L + UL \log U\right)$. The two parameters E and K depend on the cluster distribution in the dataset.

In medium-size datasets where no divide-and-conquer is needed, i.e., $p = n$, SSC-NC and LS-SSC are cubic while Lin-LS and CCC are approximately quadratic. In large-scale datasets, only algorithms designed with divide-and-conquer strategy can be run

under the constraint on RAM as well as computational power. The time complexity of LS-SSC is cubic with respect to p, while Lin-LS is almost quadratic with respect to p and cluster distribution. In fact, when n becomes very large, we can fix p in LS-SSC as an upper bound, while Lin-LS requires to synthesize the correlation matrix $n \times n$ in the coarse step. Moreover, the time complexity of Lin-LS is analyzed only on the first iteration, the cost of following iterations must be accounted. Although LS-SSC is cubic with respect to p due to Cholesky decomposition, we optimize this computation by exploiting LAPACK [110] whose implementation of Cholesky decomposition is extremely efficient. The core of our clustering framework can be found here: `https://github.com/quoctin/residual-clustering`.

3.6 Experiments

In this section, we provide experimental analyses of the proposed clustering framework. Based on real data, hyperparameters are selected and used thorough all experiments. We validate the superiority of our methodologies under intensive settings, both on medium and large-scale clustering contexts.

Dataset. All experiments are conducted on JPEG images of Dresden [3] and Vision [90]. The top-left regions of size 512×512 are cropped out for SPN extraction. We have tested diverse configurations whose quantitative details are outlined in Table 3.1 and Table 3.2, considering:

- *Cluster symmetry.* On Dresden and Vision, we create symmetric datasets containing 100 images for each camera, and asymmetric datasets containing all available images on each camera. We denote such configuration on Dresden as $\mathcal{D}_c^a$ $\mathcal{D}_c^s$, and on Vision as $\mathcal{V}_c^a$ $\mathcal{V}_c^s$, where a and s stand for *symmetric* and *asymmetric*, respectively, and c is the number of cameras.

- *Multiple instances of the same model.* On Dresden, we create datasets containing 5 camera instances of each camera model. Combining with cluster symmetry, we obtain symmetric and asymmetric datasets of this configuration as $\mathcal{D}_c^{sm}$ and $\mathcal{D}_c^{am}$.

- *Number of cameras.* In medium-size datasets, we first select $c = 5$, and incrementally add 5 cameras till $c = 20$.

- *Large-scale clustering.* On Dresden, we first select $c = 30$, and incrementally add 5 cameras till the maximum $c = 74$, considering all cameras. Since Vision is smaller than Dresden, we start with $c = 21$ and incrementally add 3 cameras till $c = 33$. Such configurations on Dresden and Vision are respectively denoted as $\mathcal{LD}_c^a$ and $\mathcal{LV}_c^a$. All these configurations include cameras of same models.

Table 3.1: Testing configurations on medium-size datasets.

Configuration		# cameras		# models		# images	
Dresden	Vision	Dresden	Vision	Dresden	Vision	Dresden	Vision
$\mathcal{D}_5^s$	$\mathcal{V}_5^s$	5		5		500	
$\mathcal{D}_{10}^s$	$\mathcal{V}_{10}^s$	10		10		1000	
$\mathcal{D}_{15}^s$	$\mathcal{V}_{15}^s$	15		15		1500	
$\mathcal{D}_{20}^s$	$\mathcal{V}_{20}^s$	20		20		2000	
$\mathcal{D}_5^a$	$\mathcal{V}_5^a$	5		5		1089	1041
$\mathcal{D}_{10}^a$	$\mathcal{V}_{10}^a$	10		10		1954	2110
$\mathcal{D}_{15}^a$	$\mathcal{V}_{15}^a$	15		15		3031	3208
$\mathcal{D}_{20}^a$	$\mathcal{V}_{20}^a$	20		20		4186	4435
$\mathcal{D}_5^{sm}$	–	5	–	1	–	500	–
$\mathcal{D}_{10}^{sm}$	–	10	–	2	–	1000	–
$\mathcal{D}_{15}^{sm}$	–	15	–	3	–	1500	–
$\mathcal{D}_{20}^{sm}$	–	20	–	4	–	2000	–
$\mathcal{D}_5^{am}$	–	5	–	1	–	855	–
$\mathcal{D}_{10}^{am}$	–	10	–	2	–	2663	–
$\mathcal{D}_{15}^{am}$	–	15	–	3	–	3558	–
$\mathcal{D}_{20}^{am}$	–	20	–	4	–	4540	–

Performance metric. We report performance in $\mathcal{F}$-measure and Adjusted Rand Index (ARI). In the presence of outliers (unclustered SPNs), we follow [2] and treat outliers differently in the computation of True Positive ($\overline{TP}$) and False Positive ($\overline{TP}$). Specifically,

- True Positive ($\overline{TP}$): the number of image pairs from the same cluster which are assigned to the same cluster, *excluding outliers*.
- False Positive ($\overline{FP}$): the number of image pairs from different clusters which are assigned to the same cluster, *excluding outliers*.
- True Negative (TN): number of image pairs from different clusters which are assigned to different clusters.
- False Negative (FN): number of image pairs from the same cluster which are assigned to different clusters.

$\mathcal{F}$-measure is computed based on precision ($\mathcal{P}$) and recall ($\mathcal{R}$):

Table 3.2: Testing configurations on large-scale datasets.

Configuration		# cameras		# images	
Dresden	Vision	Dresden	Vision	Dresden	Vision
$\mathcal{LD}^a_{30}$	$\mathcal{LV}^a_{21}$	30	21	6596	4397
$\mathcal{LD}^a_{35}$	$\mathcal{LV}^a_{24}$	35	24	7538	5051
$\mathcal{LD}^a_{40}$	$\mathcal{LV}^a_{27}$	40	27	8545	5773
$\mathcal{LD}^a_{45}$	$\mathcal{LV}^a_{30}$	45	30	9635	6377
$\mathcal{LD}^a_{50}$	$\mathcal{LV}^a_{33}$	50	33	10765	7070
$\mathcal{LD}^a_{55}$	–	55	–	11673	–
$\mathcal{LD}^a_{60}$	–	60	–	12729	–
$\mathcal{LD}^a_{65}$	–	65	–	13995	–
$\mathcal{LD}^a_{70}$	–	70	–	14915	–
$\mathcal{LD}^a_{74}$	–	74	–	15677	–

$$\mathcal{P} = \frac{\overline{TP}}{\overline{TP}+\overline{FP}}, \quad \mathcal{R} = \frac{\overline{TP}}{\overline{TP}+FN}, \quad \mathcal{F} = 2 \cdot \frac{\mathcal{P} \cdot \mathcal{C}}{\mathcal{P}+\mathcal{C}}.$$

Rand Index (RI) and ARI are computed as:

$$\text{RI} = \frac{\overline{TP}+TN}{\overline{TP}+TN+\overline{FP}+FN}, \quad \text{ARI} = \frac{\text{RI}-\mathbf{E}[\text{RI}]}{1-\mathbf{E}[\text{RI}]},$$

where $\mathbf{E}[\text{RI}]$ is the expected value of RI and is computed based on the expected value of $\overline{TP}$ and TN.

$$\mathbf{E}[\text{RI}] = \frac{\mathbf{E}[\overline{TP}]+\mathbf{E}[TN]}{\overline{TP}+TN+\overline{FP}+FN}.$$

More details of $\mathcal{F}$-measure and ARI in the presence of outliers can be found in Appendix 7.3. When the number of outliers is zero, $\mathcal{F}$-measure and ARI become canonically defined.

For comparing the number of clusters discovered by each algorithm, we follow [2] to report the ratio L_p/L_g where L_p refers to the number of predicted clusters and L_g the number of ground-truth clusters. Differently to [2] where L_p only accounts for *unique* predicted clusters, i.e., $L_p \leq L_g$, it is possible in our evaluation that $L_p/L_g > 1$ if an algorithm overestimates, or $L_p/L_g < 1$ if under-estimating the number of ground-truth clusters.

Performance comparison. We compare the results of the proposed methodologies with the state of the art. Tests have been done also with hierarchical clustering [41],

Markov Random Field [38], and Spectral Clustering with Normalized Cut criterion [44], but for the sake of space and readability we only present comparisons with the following top performing works:

- *Multiclass Spectral Clustering (MSC)* [43]. A star graph is built with 5 nearest neighbors, as suggested in [43].
- *Lin's Large-Scale (Lin-LS) method* [2]. Lin-LS is implemented with all parameters recommended from [2]: the correlation matrix of compressed SPNs (256×256) are binarized by threshold 0.008, while the correlation matrix of original-size SPNs (1024×1024) are binarized by threshold 0.005. In order to take divide-and-conquer strategy into effect on medium-size datasets, each dataset is split into two equal batches and only one is loaded at once.
- *Correlation Clustering with Consensus (CCC)* [46]. Results of CCC are acquired from the implementation provided by the authors. No parameter needs to be specified.
- *Sparse Subspace Clustering (SSC)* [6]. To validate the effectiveness of non-negativity constraint, we compare with ordinary SSC which is implemented similarly to SSC-NC but without the non-negativity constraint.

We analyze the performance of LS-SSC to verify its adaptation on medium-size and large-scale datasets. To simulate divide-and-conquer on medium-size datasets, LS-SSC splits each dataset into two equal batches and only one is loaded at once in the same manner as Lin-LS.

Under large-scale datasets, LS-SSC is compared only to Lin-LS since these methods are particularly designed for large-scale contexts. One matter of clustering on large-scale datasets is the lack of memory. Since only a limited number of SPNs can be allocated on RAM, we fix this bound to 4000 ($\approx$ 4 GBs are required to store SPNs).

Due to some randomization used in MSC, CCC, Lin-LS and LS-SSC, those methods are run 10 times, and the average scores are reported.

3.6.1 Settings

In order to select a number of parameters required by our methodologies we collect a dataset, obviously different from the test one. From RAISE dataset [111] we extract 200 raw images from Nikon D90 and 250 from D7000, and perform JPEG compression (quality factor 98). Since there are only 76 raw images of Nikon D40, we leave them

out and instead select 300 JPEG images (default JPEG quality setting) from an external Canon 600D. We refer to this dataset as $\mathcal{D}_{\text{dev}}$ including 750 images from 3 cameras.

Selecting η. η is the augmented Lagrangian hyperparameter which stands for how much penalty added in order to enforce the equality $\mathbf{Z} = \mathbf{V}$. This parameter partially decides the convergence speed of CONSTRAINED_LASSO. Small η means slow convergence but with high accurate solutions, while large η accelerates convergence speed but results in modest accurate solutions. Since sparse representation learning is followed by a clustering procedure, solutions with modest accuracy are sufficient. On $\mathcal{D}_{\text{dev}}$, $\eta \in [1.0, 1.3]$ results in acceptable solutions and fast convergence. We adopt $\eta = 1.0$ in all experiments.

Selecting γ. On $\mathcal{D}_{\text{dev}}$, we vary γ in the range $[0.0001, 0.02]$ and select $\gamma = 0.0018$ that minimizes the cost function defined in Section 3.3.2 taken into account normalized cuts and eigengaps as criterions.

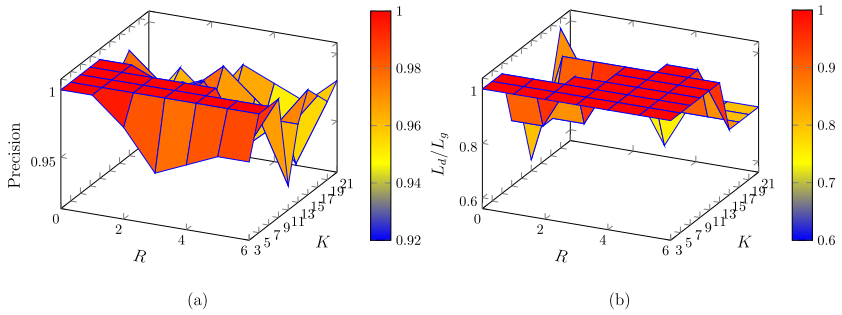


Figure 3.7: Precision and L_d/L_g with respect to diverse values of K and # recycling steps.

Jointly selecting R **and** K. In LS-SSC, the main goal of recycling is to reduce the number of undiscovered ground-truth clusters. Let us denote as L_d, L_g the number of ground-truth clusters discovered after merging phase and the number of ground-truth clusters, respectively. The strategy is to adopt the number of recycling steps R such that $L_d/L_g \to 1$ and discovered clusters are pure, namely Precision $\to 1$. Another parameter which impacts on L_d is the number of nearest neighbors K. Small K means more ground-truth clusters are likely to be discovered, otherwise only noticeably dense clusters are discovered. We conduct experiments on an asymmetric dataset from Dresden containing 5 cameras coming from different models. We split the dataset into 6 equal batches of size ≈ 182 in order to simulate splitting step. Figure 3.7 depicts precision of discovered clusters after merging step in panel (a), and the ratio L_d/L_g in panel (b). It is clear that $K = 5$ is a reasonable choice for discovering pure ground-truth clusters. From these plots one can argue that selecting $R = 0$ allows to obtain the highest precision in this case. However, it is important to remember that recycling plays an important role since it helps discover more hidden clusters. In principle, high value of R should be chosen considering

the computational complexity, but the precision is likely to drop if we run more recycling steps with big K. In large-scale contexts, we adopt $R = \lfloor B/2 \rfloor$, where B is the number of batches. In medium-scale contexts, where computational requirement is less important, we run recycling until there is no noticeable subclusters discovered.

3.6.2 Medium-Scaling Clustering

We report performance of all methods on medium-size datasets with the maximum number of images ranging from 4000 to 5000.

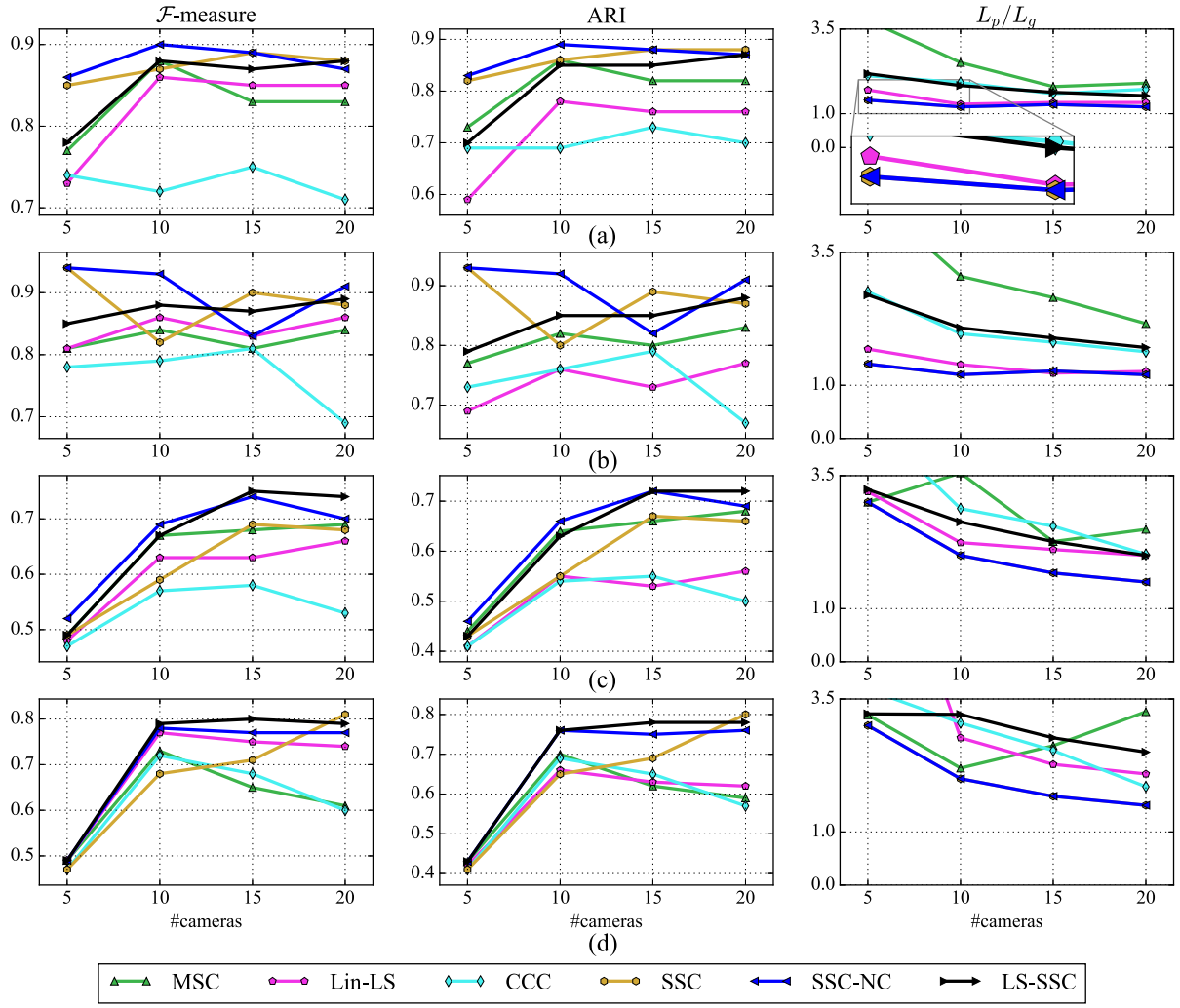


Figure 3.8: Clustering performance on medium-size datasets of Dresden: (a) symmetric: $\mathcal{D}^s_5, \mathcal{D}^s_{10}, \mathcal{D}^s_{15}, \mathcal{D}^s_{20}$ (b) asymmetric: $\mathcal{D}^a_5, \mathcal{D}^a_{10}, \mathcal{D}^a_{15}, \mathcal{D}^a_{20}$ (c) symmetric + same model: $\mathcal{D}^{sm}_5, \mathcal{D}^{sm}_{10}, \mathcal{D}^{sm}_{15}, \mathcal{D}^{sm}_{20}$ (d) asymmetric + same model: $\mathcal{D}^{am}_5, \mathcal{D}^{am}_{10}, \mathcal{D}^{am}_{15}, \mathcal{D}^{am}_{20}$. Better viewed in color.

Results on Dresden suggest that MSC performs relatively well on symmetric (in Figure 3.8 (a)) and asymmetric (in Figure 3.8 (b)) datasets. MSC applies an extra step before clustering. It is the creation of a star graph among SPNs, where noisy connections are partially eliminated. The star graph can be considered as a suboptimal sparse representation matrix of data. Differently to MSC, SSC finds a sparse representation of data by solving an optimization problem. In Figure 3.8, SSC outperforms MSC in most configurations with high $\mathcal{F}$-measure. As an improved version of SSC, SSC-NC performs equally or better than SSC in the majority of symmetric and asymmetric datasets. Balanced precision and recall are obtained, gaining high $\mathcal{F}$-measure. The number of predicted clusters L_p obtained by SSC and SSC-NC are identical, approximating well the number of ground-truth clusters L_g. Such approximation is the best among all tested algorithms.

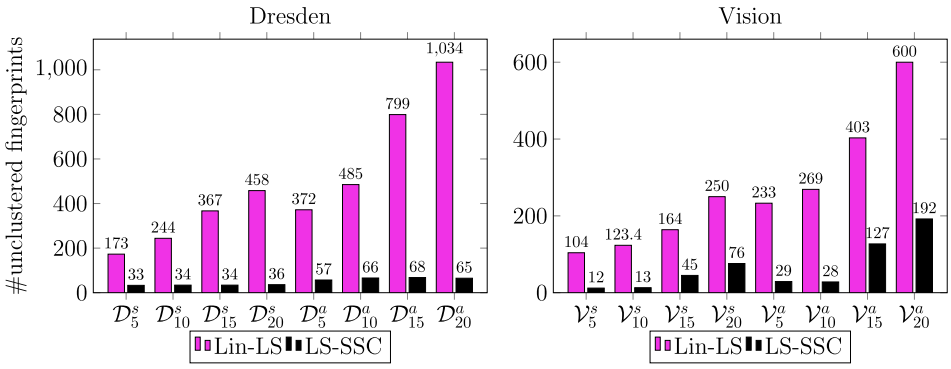


Figure 3.9: Number of unclustered SPNs on medium-size datasets of Dresden and Vision.

Although Lin-LS and LS-SSC are especially designed for large-scale datasets, they produce convincing results also on medium-size datasets. Lin-LS aims to obtain high-quality clusters of small size, resulting in high precision. Comparing to Lin-LS, LS-SSC obtains less precise clusters but the precision is still high without penalizing recall. Thanks to this balanced behavior, LS-SSC outperforms Lin-LS in terms of $\mathcal{F}$-measure and ARI. To keep precision high, both Lin-LS and LS-SSC tend to overestimate the number clusters in medium-size datasets.

Zooming into the cases where cameras of the same model share some commonalities in SPNs, this clearly introduces a certain level of ambiguity. In Figure 3.8 (c) and (d) we report results on datasets containing multiple camera models, each model with 5 camera instances. Despite the fact that all methods suffer from performance degradation, SSC-NC outperforms other methods in $\mathcal{D}^{sm}_5, \mathcal{D}^{sm}_{10}$, while LS-SSC is superior in all other configurations. In Figure 3.8 (c) and (d), the superiority of SSC-NC over SSC is evident. We argue that, in such complicated contexts where SPNs of the same camera model stay close to each other, SSC-NC can find better data representations.

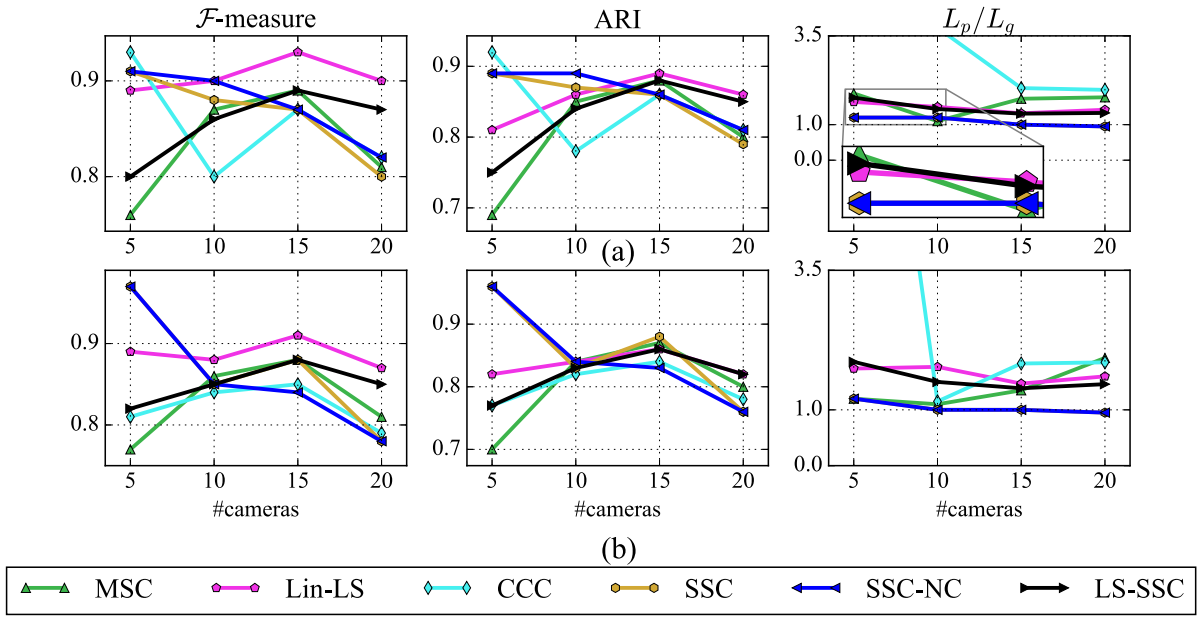


Figure 3.10: Clustering performance on medium-size datasets of Vision: (a) symmetric: $\mathcal{V}^s_5, \mathcal{V}^s_{10}, \mathcal{V}^s_{15}, \mathcal{V}^s_{20}$ (b) asymmetric: $\mathcal{V}^a_5, \mathcal{V}^a_{10}, \mathcal{V}^a_{15}, \mathcal{V}^a_{20}$. Better viewed in color.

We replicate the evaluation of all methods on medium-size datasets of Vision, see Figure 3.10. MSC, SSC-NC and LS-SSC perform on par with each other, but SSC-NC achieves more accurate estimation on the number of clusters. On the other hand, SSC-NC also obtains more accurate results than SSC in almost all configurations (7 out of 8). It seems that Lin-LS outperforms all other methods, however, we argue that its performance gain is partially due to high number of unclustered SPNs it produces. We show in Figure 3.9 the number of unclustered SPNs of LS-SSC and Lin-LS on medium-size datasets of Dresden and Vision. It is evident that Lin-LS produces more outliers than LS-SSC, thus gaining a certain advantage over precision, and then $\mathcal{F}$-measure as a consequence.

3.6.3 Large-Scale Clustering

In practice, there exist large-scale contexts where a large number of images need to be clustered. In Dresden, we conduct experiments on datasets containing 30 to 74 cameras, and the number of images exceeds 6000, while in Vision the number of cameras ranges from 21 to 33 and the number of images exceeds 4000. To the best of our knowledge, Lin-LS [2] is the only method proposed for large-scale clustering of SPNs, thus results are compared only with it.

As depicted in Figure 3.11, Lin-LS achieves high precision, which means $\overline{FP}$ is negligible. Nevertheless, in order to keep high precision a noticeable number of SPNs are

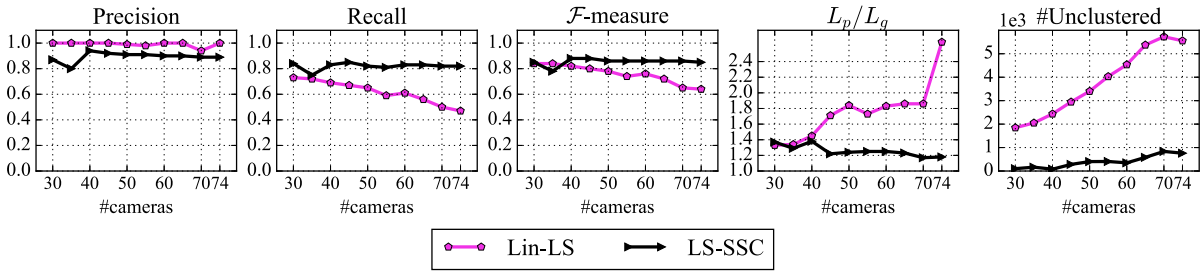


Figure 3.11: Clustering results on large-scale datasets of Dresden.

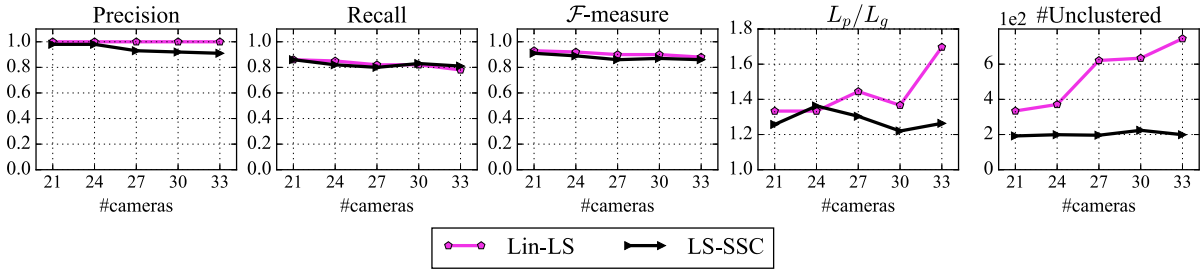


Figure 3.12: Clustering results on large-scale datasets of Vision.

not clustered. Unclustered SPNs essentially cause low recall, or equivalently high FN due to the separation of pairs belonging to the same cluster. On the contrary, LS-SSC produces less precise clusters with precision from 80% to 100%. One advantage of our method is the achievement of relatively high recall which slightly oscillates around 80%. Apart from keeping precision and recall balanced, we obtain high $\mathcal{F}$-measure. LS-SSC can cluster the whole Dresden dataset with $\mathcal{F}$-measure higher than 80% which substantially improves the 64% obtained by Lin-LS. The improvement of LS-SSC over Lin-LS should be further amplified because Lin-LS requires to access 1024×1024 SPNs in refining step while LS-SSC only works on 512×512 SPNs. Moreover, as depicted in Figure 3.11 (last panel), LS-SSC produces a higher number of clusters than the ground-truth clusters, but the ratio between the two quantities is relatively constant when the dataset size grows. Vice versa, for Lin-LS this ratio rapidly increases.

Shown in Figure 3.12 are the performance of Lin-LS and LS-SSC on Vision dataset. Lin-LS again produces highly precise clusters, but tends to overestimate the number of ground-truth clusters. The $\mathcal{F}$-measure scores of the two methods are close since unclustered SPNs are not accounted for precision computation.

In Lin-LS, the main cause of unclustered SPNs are due to the merging step. If the merging threshold is too high, small subclusters cannot be merged to form larger subclusters, and thus filtered out in the end. On the other hand, in LS-SSC a SPN is unclustered

if the correlation between it and all available cluster centroids is smaller than a threshold that was used to exclude the null hypothesis. Also for the case of large-scale datasets, we show the number of unclustered SPNs in Lin-LS and LS-SSC, see last panel of Figure 3.11, 3.12. In this scenario, it is clear that to keep precision high Lin-LS produces large number of unclustered SPNs, not comparable with unclustered SPNs in LS-SSC. The advantage of this mechanism is to reduce false alarm rate, but its downside is evident since data of interest could be ignored by the algorithm. LS-SSC provides a reasonable tradeoff allowing to cluster large-scale databases without skipping too many images which might be important for forensic analysis.

3.6.4 Analysis On The Robustness

In this section, we analyze the robustness of LS-SSC in more realistic testing configurations.

Presence of outliers. Firstly, we test the robustness of LS-SSC to outliers. We select images coming from 20 cameras of Vision, and add 50 images randomly collected from Facebook (from different entities) to make sure that they do not share the same source camera. On this dataset, LS-SSC achieves $\mathcal{F}$-measure 0.89. Remarkably, LS-SSC assigns 69 images as unclustered, in which 33 out of 50 images are truthfully outliers.

Double JPEG compression. Images taken by smartphones usually undergo a primary JPEG compression by default, and double JPEG compression once being shared via social networks. Therefore, we test the robustness of LS-SSC on images coming from 20 cameras of Vision, further compressed using `convert` tool provided by `ImageMagick`. The compression quality ranges from 50 to 95 (step 5). Results in Table 3.3 expose very reasonable and pretty stable performance of LS-SSC over different quality factors. Indeed, the algorithm is generally robust to double JPEG compression if the quality factor of the second compression is more than 65. Clustering performance starts to drop if images are aggressively compressed (quality factor smaller than 65).

Table 3.3: Numeric results of LS-SSC on double compressed images.

Metric	Quality factor									
	50	55	60	65	70	75	80	85	90	95
$\mathcal{P}$	0.79	0.81	0.84	0.88	0.90	0.93	0.94	0.95	0.93	0.96
$\mathcal{R}$	0.58	0.61	0.66	0.72	0.77	0.80	0.83	0.86	0.85	0.88
$\mathcal{F}$	0.67	0.70	0.74	0.79	0.83	0.86	0.88	0.90	0.89	0.92
ARI	0.65	0.68	0.73	0.78	0.82	0.85	0.87	0.89	0.88	0.88

Images from social networks. Images acquired from online social networks are

invaluable resources for forensic investigations. Unfortunately, once being uploaded and shared online images undergo strong processing such as JPEG compression and resizing which substantially remove forensic traces. By taking images uploaded via Facebook in high-quality and low-quality mode from Vision [90], we test LS-SSC on large-scale settings where the number of cameras ranges from 21 to 33 with step 3 and report results in Figure 3.13 in terms of $\mathcal{F}$-score. Unsurprisingly, performance of LS-SSC fairly decreases correspondingly to the quality of images. On high-quality upload, $\mathcal{F}$-score is about 60%, while this number decreases to 50% on low-quality upload. We also observe that in low-quality upload, LS-SSC produces more unclustered images than high-quality upload. This is in fact an expected behavior as low-quality images are marked as outliers. On another hand, it has been confirmed from [46] that SPN-based clustering methods are susceptible to images downloaded from Facebook since the signal of interest is significantly distorted.

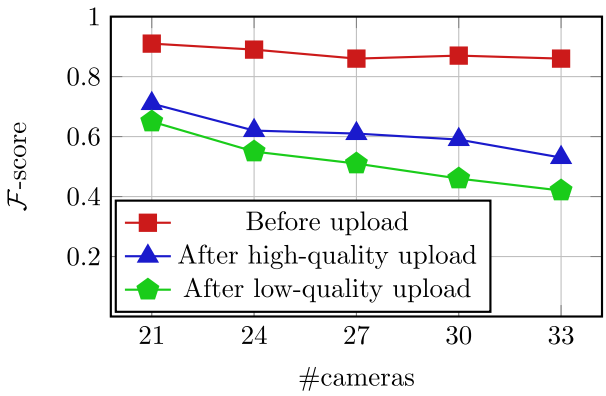


Figure 3.13: Performance of LS-SSC on Vision images before uploaded, after uploaded in high-quality and low-quality mode.

Different SPN sizes. Next, we validate the robustness of LS-SSC to different sizes of SPN. We pick the same set of images used in previous experiment, but crop the top-left region to 4 different sizes: $256\times256, 512\times512, 768\times768, 1024\times1024$. The hyperparameter γ is also re-estimated on the development set where images are cropped to similar sizes. The values of γ for each of corresponding size are $0.0045, 0.0018, 0.0012, 0.0008$. In Table 3.4, the performance generally improves if larger-size SPNs are used. Nevertheless, the results also suggest that using SPN sizes larger than 512×512 is not the key for the success of LS-SSC. Indeed, using 768×768 does not gain any improvement over 512×512 SPNs, and using 1024×1024 SPNs brings only a minor improvement.

Few images per camera. In some specific contexts, forensic analysts might face with databases where the number of cameras is higher than the average number of images

Table 3.4: Numeric results of LS-SSC on SPNs of different sizes.

Metric	SPN size			
	256×256	512×512	768×768	1024×1024
$\mathcal{P}$	0.85	0.92	0.92	0.89
$\mathcal{R}$	0.84	0.84	0.85	0.90
$\mathcal{F}$	0.84	0.88	0.88	0.89
ARI	0.83	0.87	0.87	0.88

acquired by each camera (one camera per each model). To simulate such context, we start with an original set of 20 cameras selected from Dresden. The number of images on each camera alternatively ranges from 10 to 50 (step 10). For each image, we crop at 50 different positions, ending an augmented set of images coming from 50 cameras. We fianlly obtain a dataset of 12000 images of 400 cameras. It is a challenging dataset since the number of cameras is high, while the number of images for each camera is much lower. LS-SSC assigns images into 258 clusters, and 469 images remain unclustered. Obviously, many small-size clusters are hard to be discovered due to random splitting.

It is acknowledged in [2] that Lin-LS is especially designed to cope with such scenarios. However, such capability comes at a cost of discarding many outliers, which might leave images of interest out of consideration. Lin-LS assigns images into 892 clusters, while 4083 images remain unclustered. We obtain an $\mathcal{F}$-measure 47% in this dataset, while Lin-LS achieves 52%, at a price of a 10 times larger number of unclustered images.

In this scenario, LS-SSC performs not very well, but this is somehow inherently defined in the method itself. Indeed, we know from the theory that learning sparse representation of camera SPNs requires sufficient number of images per camera. Without this assumption, the algorithm might learn inexact representations which usually result in high $\overline{FP}$.

3.6.5 Running time analysis

We measure the running time of SSC-NC and LS-SSC on Dresden images, where the number of cameras ranges from 10 to 70. To observe the running time of SSC-NC we assume RAM is sufficient to catch all SPNs of 70 cameras, and allows to solve the optimization in Equation (3.5). Figure 3.14 (a) reveals the fact that LS-SSC requires higher I/O cost due to extra reading/writing operations. SSC-NC, on the other hand, requires much higher computational cost, which are critical in practical usages. For LS-SSC, it takes approximately 1 hour and 20 minutes to cluster the whole Dresden dataset. In the case of limited RAM, LS-SSC requires more I/O time while SSC-NC cannot be operated.

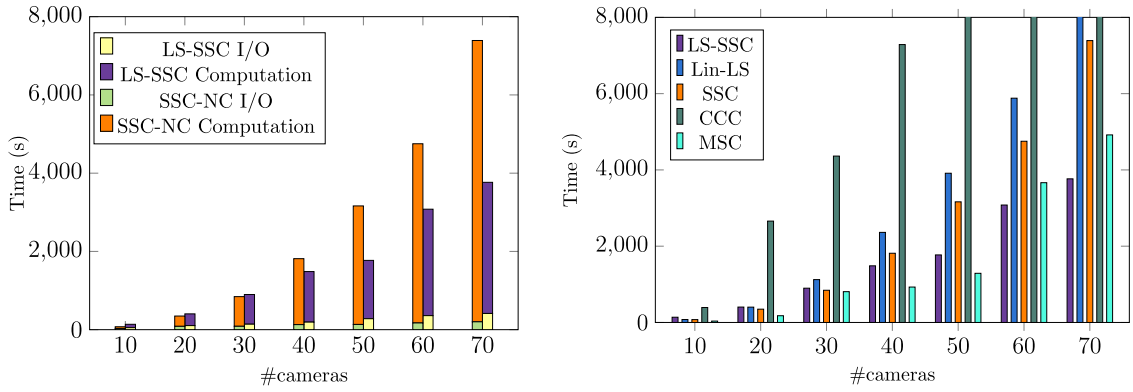


Figure 3.14: (a) Running time of SSC-NC and LS-SSC. (b) Running time of all methods.

The total running time of all methods are shown in Figure 3.14 (b) [1]. LS-SSC exposes its computational advantage when the number of images significantly increases. While Lin-LS is claimed to take approximately 45 minutes to cluster the whole Dresden [2], our implementation yields inferior performance. The computational time of CCC is quite slow even though we use the implementation of the authors. While SSC, MSC appear to be acceptable, we emphasize again that when RAM is limited, they cannot be operated.

3.7 Conclusions

In this chapter, two clustering methodologies are proposed for medium-scale and large-scale contexts. The proposed methodologies seek for better representations of SPNs in order to improve clustering accuracy. Since SPNs are high-dimensional and noisy, traditional measure like normalized correlation results in severe inter-class correlations, making the affinity matrix ambiguous. By leveraging Sparse Subspace Clustering (SSC), whose principle is to find sparse linear relationships among data points, we successfully design a novel clustering framework for clustering SPNs both in medium-scale and large-scale contexts. Despite many advantages of our framework, we anticipate certain scenarios that is hard to deal with:

- **Few images per camera**. In high dimensional vector space, a data point must be reconstructed by a large number of linearly independent data points. Otherwise,

[1] CPU time of execution strongly depends on the implementation and may mislead the reader in the presence of suboptimal implementations. Also, optimization or parallelization of the code may affect the execution time (e.g., the code of CCC [46] provided by the authors contains parallel `for` loops). Since we cannot guarantee a fair evaluation of execution times, in particular for competing methods, we prefer to provide a detailed computational complexity analysis (Section 3.6.5).

we cannot ensure the small reconstruction error. This theoretical weakness limits SSC-based methods to cases where sufficient images per camera are available.

- **Pixel misalignment**. Since SPNs are caused by imperfections of camera sensors, two images that come from the same camera and are misaligned with respect to pixel position, will not well correlated. This problem is well-known in the literature and to the best of our knowledge, has not been seriously addressed. Our methodologies are not a solution to this problem because we work on the same data. More unfortunately, images available online usually undergo geometric transformations which cause pixel misalignment. Our preliminary experiments on Facebook images has confirmed the inferior performance. Such phenomena have been observed also in [46].

Recent efforts have shown that SPNs are compressible [112, 113, 114, 115, 116], and these findings are interesting to be considered in unsupervised context. A potential extension of this work is to conquer the high dimensionality in SPNs by resorting to dimensionality reduction methods that expectedly preserve the underlying subspace structure. Another research question could be investigated in the near future is how features extracted by DNNs [25, 26, 117] are effective in this clustering problem.

Chapter 4

Identifying Social Network Origin of Images

Nowadays, Social Network (SN) are privileged channel for diffusion of digital images. Most of SN platforms employ strong processing operations, e.g., compression and resizing, in order to optimize the storage cost, transfer bandwidth as well as display quality. On one side, such strong processing significantly destroys forensic traces, hindering image forensic investigations. On another side, processing done by SNs leave their own peculiars on images which enable the distinction of SN origin of images. Being able to identify a posteriori the SN origin of images brings useful evidences to forensic analyst in certain scenarios.

In this chapter, we discuss the exploitation of traces jointly from Discrete Cosine Transformation (DCT) coefficients and metadata for the identification of Public Social Network (PSN) platforms as well as Instant Messaging Apps (IMAs) as they characterize JPEG compression and resizing. Furthermore, our effort is also spent on understanding to which extent the chain of sharing platforms can be identified when the image is circulated across multiple platforms.

Acknowledgement

I would like to thank Dr. Cecilia Pasquini from University of Innsbruck, Prof. Roberto Caldelli, and Dr. Irene Amerini from University of Florence for their kindly support during this study.

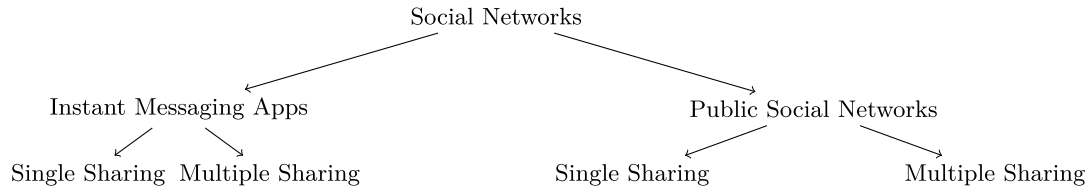


Figure 4.1: Hierarchy of our considerations in the identification of social network origin.

It is popular nowadays that images after acquired by a smart device are shared immediately through (online) Social Network (SN). Due to the complex implementations of modern SNs, this study will investigate different scenarios, see Figure 4.1. Firstly, we draw a boundary between IMAs and PSNs, the two most common forms of SNs but intrinsically distinctive. PSNs enable users to upload and share user-generated content which can be viewed and interacted publicly by many users, on the contrary IMAs are designed for private communications. PSNs are more popular on Web environment, while IMAs are typically made for mobile devices. In both cases, images can be transferred once or through many hops. We refer to them as *single sharing* and *multiple sharing* scenarios. Further information on how these operations are done on IMAs and PSNs will be provided in Section 4.5.1. To this end, our considerations are oriented to single sharing and multiple sharing scenarios on IMAs and PSNs.

Before storing images, SN platforms often employ their own policy including different operations like resizing and JPEG compression in order to optimize the storage cost, transfer bandwidth as well as display quality. Such policies are in fact neither known, nor fixed. Under certain conditions, the operations performed by SN platforms leave forensic traces which can be exploited to a posteriori identify the SN platform(s) an image has been circulated through. In the following section, we recall the JPEG compression process and possible traces that can be investigated.

4.1 Traces of JPEG Compression

The vast majority of digital images are in JPEG format, a popular compressed format which allows to store images in smaller files by retaining only the amount of information necessary to maintain their visual quality. It plays a key role in the analysis of SN origin, since JPEG compression can be applied according to several tunable parameters, that are stored in the header of the final JPEG file.

The JPEG compression scheme is depicted in Figure 4.2. In JPEG compression standard [118], an image in three channel color space (RGB) is first converted to luminance/chrominance channel space (YCbCr). The two chrominance channels (CbCr)

are typically subsampled by a factor of two relative to the luminance channel (Y). Each channel is partitioned into non-overlapping 8×8 blocks, and pixel values within each block are converted into $[-128, 127]$. Each block is subsequently transformed into 8×8 coefficients using 2-D Discrete Cosine Transformation (DCT). Let $F_c(u, v)$ be a DCT coefficient on the spatial frequency (u, v) and channel c. For simplicity, we consider only luminance channel and c is omitted as chrominance channels are often subsampled by the factor of two or four and subject to stronger quantization, they contain less information than luminance channel [55].

$$F(u,v) = a_{u,v} \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right),$$

where $a_{u,v}$ is the scaling factor, and $f(\cdot)$ is the pixel value of luminance channel (after converted), $0 \leq u, v \leq 7$.

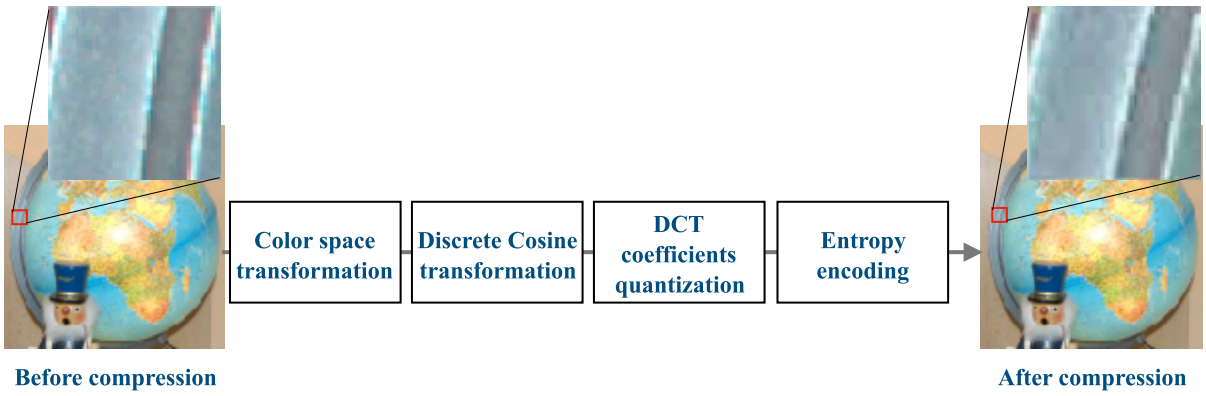


Figure 4.2: Pipeline of the JPEG compression scheme. The effect of JPEG compression can by observed in the resulting image. By zooming closer to one portion of the image, we can notice the artifacts of 8×8 blocks. Image taken from Dresden dataset [3].

In the next step, each DCT coefficient is quantized by an amount $q(\cdot)$ (quantization step) depending on the spatial frequency u, v.

$$\hat{F}(u,v) = \text{round}\left(\frac{F(u,v)}{q(u,v)}\right). \tag{4.1}$$

This step is the main cause of information loss. For low compression rates, $q(u, v)$ tends to 1. For higher compression rates, $q(u, v)$ tends to increase. The quantization for low frequency components is typically less than for high frequency components since low frequency components contain most of visual information and need to be retained. The quantization for luminance channel is also typically less than for chrominance channels.

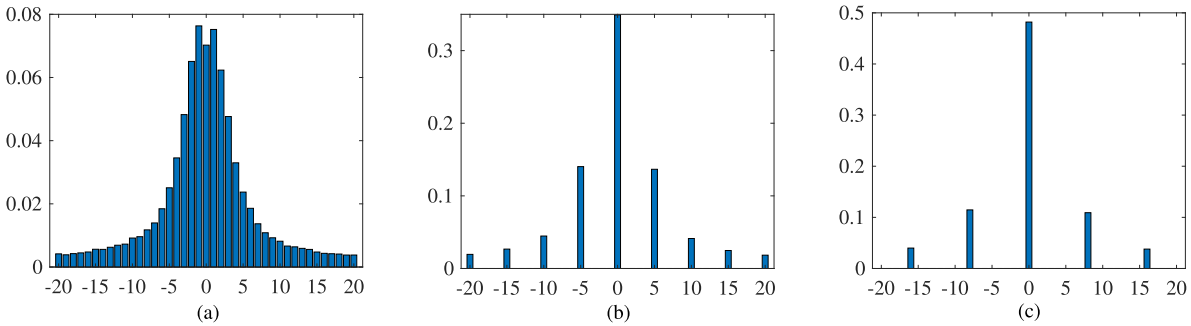


Figure 4.3: (a) $H_{1,0}$ associcated with $q(1,0) = 1$, (b) $H_{1,0}$ associcated with $q(1,0) = 5$, (c) $H_{1,0}$ associcated with $q(1,0) = 8$.

After quantization, quantized DCT coefficients are entropy encoded to reduce the storage size. Entropy encoding encodes frequent values by shorter code length, and less frequent values by longer code length. Entropy encoding is lossless, and it works well due to the fact that after quantization majority of $\hat{F}(u,v)$ are zeros (frequent value).

It is well-known that SN platforms employ JPEG compression for uploaded images, and the experimental investigation in [55] confirms this fact. Therefore, the identification of SN is not the verification whether JPEG compression has been performed, but more about *how it has been done*. In JPEG compression pipeline, DCT coefficient quantization is the step that can be mostly customized by SN platforms.

Histogram of DCT coefficients reveals JPEG compression artifacts [18]. Let $H_{u,v}$ the histogram of $\hat{F}(u,v)$ over all 8×8 blocks of luminance channel. As rounding operation in Equation (4.1) is many-to-one function, single JPEG compression with different quantization steps $q(u,v)$ will result in distinctive histograms. Figure 4.3 represents $H_{1,0}$ associated with (a) $q(1,0) = 1$, (b) $q(1,0) = 5$, and (c) $q(1,0) = 8$. Specifically, higher quantization step results in *sparser* histogram since many values are rounded to the same bin. The choice of quantization steps is the main source of variation, and it therefore can be exploited for SN origin identification. Most of acquisition devices select JPEG as the default format, and images are subjected to the second JPEG compression once they are uploaded to SN platforms. If an image is doubly compressed using the quantization step $q_1(u,v)$ followed by $q_2(u,v)$, the quantized DCT coefficient $\hat{F}'(u,v)$ of the final JPEG file is expressed as:

$$\hat{F}'(u,v) = \text{round}\left(\frac{q_1(u,v)}{q_2(u,v)} \times \text{round}\left(\frac{F(u,v)}{q_1(u,v)}\right)\right). \tag{4.2}$$

The presence of double JPEG compression can be detected also by observing the histogram of $\hat{F}'(u,v)$, denoted by $H'_{u,v}$. The interplay of $q_1(u,v)$ and $q_2(u,v)$ decides the appearance of $H'_{u,v}$.

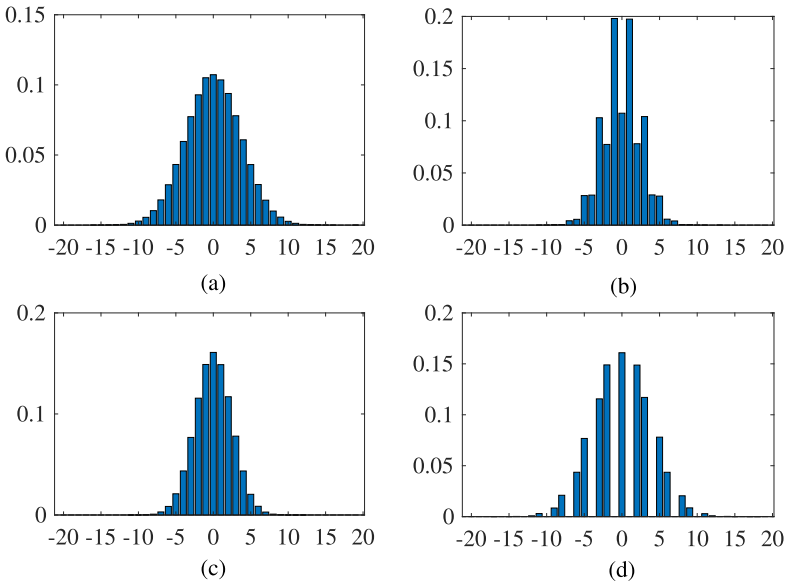


Figure 4.4: (a) $H_{1,0}$ associcated with $q_1(1,0) = 2$, (b) $H'_{1,0}$ associcated with $q_1(1,0) = 2$ and $q_2(1,0) = 3$, (c) $H_{1,0}$ associcated with $q_1(1,0) = 3$, (d) $H'_{1,0}$ associcated with $q_1(1,0) = 3, q_2(1,0) = 2$.

In Figure 4.4 (a), the original image is firstly compressed with the quantization step $q_1(1,0) = 2$, and in (b) it is doubly compressed with $q_2(1,0) = 3$ respectively. In Figure 4.4 (c), the original image is firstly compressed with the quantization step $q_1(1,0) = 3$, and in (d) it is doubly compressed with $q_2(1,0) = 2$. It is worth to note that JPEG quantization and dequantization are simulated to obtain Figure 4.4 instead of real implementation. The reason is that real implementation introduces some sources of errors: color conversion, truncation of the values to eight bit integers, DCT inverse transformation; and therefore it is hard to observe the clean effects of double JPEG compression. As we can see from Figure 4.4 (b), when $q_2(1,0) > q_1(1,0)$ the histogram shrinks towards 0 and some bins contain more samples than neighboring bins. On the other hand, Figure 4.4 (d) shows the case $q_2(1,0) < q_1(1,0)$ the histogram expands to wider bin ranges and some bins are empty.

Detection of double JPEG compression has been vastly studied, most of them exploit the artifacts from histogram of DCT coefficients. Remarkably, in [18, 119] the presence of double JPEG compression can be detected by looking at periodic patterns on the histogram in frequency domain. Differently, [120] detects double compression by training a SVM classifier directly on absolute values of histogram bins. Not only detecting the presence of double JPEG compression, [78] attempts to estimate the first quantization table by training multiple SVMs on the same feature set as [120]. All of these methods

consider rather restricted scenarios where the second quantization table is different from the primary one. Due to errors during JPEG compression, i.e., quantization error, truncation error and DCT inverse transformation, the two successive compressions with the same quantization table could produce two pixel-wise different images. [121] exploits the number of unique DCT coefficients between the two compressions to design a decision rule for the detection of double JPEG compression. In fact, the number of unique DCT coefficients is also encoded in the histogram of DCT coefficients.

Given the distinctive power of the aforementioned histogram, it is potential for the identification of SN platforms since each platform employs its own parameters in JPEG compression pipeline. Unfortunately, such methods are feasible only on limited cases:

- *Aligned double JPEG compression*: the grids of successive compressions are aligned.
- *No resizing between the two compressions*: the image sizes between two successive compressions are equal.

For that reason, the exploitation of other possible traces apart from histogram of DCT coefficients is useful. The next section presents distinctive attributes from metadata that can be opted for the identification of SN platforms.

4.2 Traces from Metadata

Metadata is simply data about data. In digital images, metadata contains useful information for forensic investigation and they should not be overlooked in every investigation process. For instance, EXIF (Extended File Information) contains information about the camera that took the picture, date/time of the acquisition. Below is a typical EXIF header in human readable format [122]:

`File name`: 0805-153933.jpg
`File size`: 463023 bytes
`File date`: 2001:08:12 21:02:04
`Camera make`: Canon
`Camera model`: Canon PowerShot S100
`Date/Time`: 2001:08:05 15:39:33 Resolution: 1600 × 1200
`Flash used`: No
`Focal length`: 5.4mm (35mm equivalent: 36mm) CCD Width: 5.23mm
`Exposure time`: 0.100 s (1/10)
`Aperture`: f/2.8
`Focus Dist.`: 1.18m

`Metering Mode`: center weight
`Jpeg process`: Baseline

Such piece of information is useful, but obviously does not characterize the SN platform. Work in [123] shows that apart from EXIF header, other information from header of a JPEG file characterizes the JPEG compression process, which is highly distinctive across cameras:

- `Image dimensions`: two integers identifying the image size. This information is useful to discriminate different sensor resolution.
- `Quantization tables`: the set of three 8×8 quantization tables corresponding to luminance (Y), chrominance (Cb), and chrominance (Cr). The entries on each table are quantization steps, used to quantize DCT coefficients before entropy encoding.
- `Huffman code`: the six sets of 15 values corresponding number of codes of length $1, \ldots, 15$: each of three channels require two codes, one for DC and one for AC coefficients. In fact, it is very common that Cb and Cr share the same tables.

Besides, some additional parameters can be also retrieved:

- `Component information`: the set of 18 integer numbers describing auxiliary information on each luminance/chrominance channel. Below is a typical table for such values:

component_id	h_samp_factor	v_samp_factor	quant_tbl_no	dc_tbl_no	ac_tbl_no
1	2	2	1	1	1
2	1	1	2	2	2
3	1	1	2	2	2

  The first row contains information of the luminance channel, i.e. this channel is up-sampled by a factor of 2 (relative to chrominance channel) on horizontal and vertical axis; its quantization table index is 1; coding table index is 1 for both DC and AC coefficients. The information of chrominance channels in the second and the third row can be read in a similar manner.

- `Progressive mode`: boolean value indicating whether progressive mode is applied.
- `Optimized coding`: boolean value indicating whether optimized Huffman tables are used.

In this study, we investigate two kinds of feature set: i) histogram of DCT coefficients and metadata, the so-called hand-crafted features, and ii) deep features extracted by

Convolutional Neural Networks (CNNs). The next section focuses on hand-crafted feature extraction.

4.3 Exploitation of Hand-Crafted Features

Hand-crafted features composes two main kinds of features:

- Histogram of DCT coefficients, denoted by SignalFea.
- Metadata-based features, denoted by MetaFea.

As mentioned in Section 4.1, DCT coefficients from chrominance (CbCr) contain less statistical information than of luminance (Y) channel since chrominance channels are typically subject to subsampling and stronger quantization. We therefore extract SignalFea on luminance channel only. For each 8×8 block of DCT coefficients, we select a set of quantized DCT coefficients $\hat{F}(u, v)$ of the last JPEG compression (we do not assume to know the number of JPEG compression) where (u, v) are limited to the set of 9 low spatial frequencies:

$$LF = \{(0,1), (1,0), (2,0), (1,1), (0,2), (0,3), (1,2), (2,1), (3,0)\}.$$

The DC frequency $(u = 0, v = 0)$ is omitted. The dequantized DCT coefficients $\tilde{F}(u, v)$ are then computed by:

$$\tilde{F}(u, v) = \hat{F}(u, v) \times q(u, v).$$

The histograms of $\tilde{F}(u, v)$ over all blocks are accumulated, normalized by number of blocks, and denoted by $H_{u,v}$. To avoid the explosion in number of features, we only retain histogram bins in $[-20, 20]$ (bin width is 1) which significantly capture statistical information of DCT coefficients. The final feature set is the concatenation of all $H_{u,v}$ with $(u, v) \in LF$. The length of SignalFea is $41 \times |LF| = 369$.

On the other hand, MetaFea is extracted from the JPEG header. We list here the data extracted the corresponding number of features in brackets (total number 152):

- *Quantization tables* (128): extracted for luminance channel and chrominance channel, they contain the quantization steps and reflect the quality factor of the (last) JPEG compression.
- *Huffman encoding tables* (2): number of encoding tables used for AC and DC component.
- *Component information* (18): for the components Y, Cb, Cr, 6 features describe component id, horizontal/vertical sampling factors, quantization table index, AC/DC coding table indices.

- *Optimized coding and progressive mode* (2): Binary values indicating the use of optimized coding and progressive mode.

- *Image size* (2): image dimensions.

Hand-crafted features, denoted by HanCrFea, is the concatenation of SignalFea and MetaFea, ending up 521 features. The evaluation of hand-crafted feature is carried out under three well-known classifiers: LR (Logistic Regression), SVM (linear Support Vector Machine with penalty parameter 1) and RF (Random Forest with 100 estimators).

4.4 Exploitation of Deep Features

By the recent advance of deep learning and big amount of data available, several forensic problems can be solved at better performance: tampering detection [124, 125, 126, 71, 127, 26], device identification [25], device linking [128], double JPEG compression [129]. In particular, in [56] a CNN-based approach has been proposed for the identification of SN, showing the potential of deep learning approaches for this particular problem.

In this section, we investigate further a CNN-based solution for end-to-end classification, and introduce the combination of MetaFea and deep features extracted from CNN trained on large amount of collected data.

4.4.1 CNN Architecture and Training Tactics

We propose a new CNN architecture that is able to work directly on DCT coefficients as input. Our CNN is named as P-CNN, and it is sketched in Figure 4.5, which takes as input a DCT coefficient map of $\mathfrak{B} \times \mathfrak{B}$ ($\mathfrak{B}$ is multiple of 8) and outputs probability scores over class labels, i.e. P-CNN Predictions.

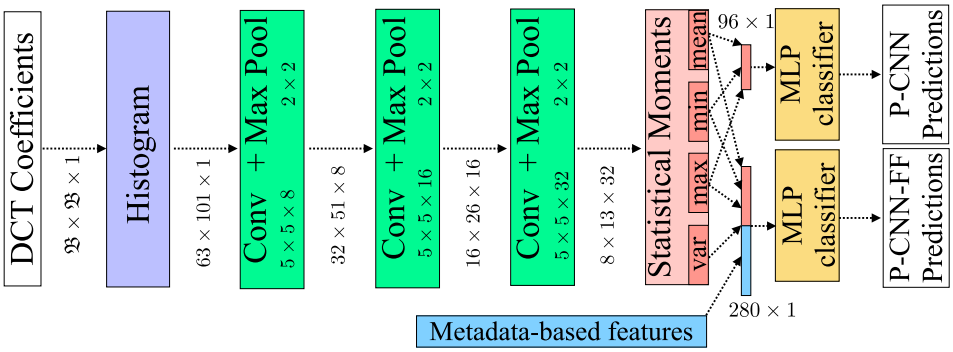


Figure 4.5: Architecture of P-CNN and P-CNN-FF.

A typical way to learn statistics of DCT coefficients on each spatial frequency is computing their histogram; and this feature has been successfully incorporated into CNNs [130, 129]. Denote $\hat{F}_{i,j}(u,v)$ the quantized DCT coefficient in the (i,j)-th block $(1 \leq i,j \leq \mathfrak{B})$. In principle, bin x of histogram $H_{u,v}$ can be computed by an indicator function $\mathbb{I}(\cdot)$:

$$H_{u,v}(x) = \frac{64}{\mathfrak{B}^2} \sum_{i=1}^{\mathfrak{B}/8} \sum_{j=1}^{\mathfrak{B}/8} \mathbb{I}_x(\hat{F}_{i,j}(u,v)),$$

where:

$$\mathbb{I}_x\left(\hat{F}_{i,j}(u,v)\right) = \begin{cases} 1, & \hat{F}_{i,j}(u,v) = x \\ 0, & \hat{F}_{i,j}(u,v) \neq x \end{cases}.$$

A limitation of indicator function in end-to-end CNNs is its non-smoothness, i.e. zero derivative everywhere except x. It prevents back-propagating errors during training. In [130], the authors proposed to replace indicator function by a Gaussian activation centered at x. This solution allows the back flow of gradients, but it requires evaluating Gaussian activation multiple times – it is the number of bins by number of DCT coefficients in our problem, and more importantly it has not been supported yet by common deep learning frameworks. Because of its complication, it is suitable to histograms with small number of bins.

Work in [129] proposes another alternative, where a sigmoid activation is used as an approximation of indicator function. Specifically, bin x of histogram $H_{u,v}$ is computed, firstly by means of an accumulative histogram $A_{u,v}$:

$$A_{u,v}(x) = \frac{64}{\mathfrak{B}^2} \sum_{i=1}^{\mathfrak{B}/8} \sum_{j=1}^{\mathfrak{B}/8} \sigma\left(\zeta(\hat{F}_{i,j}(u,v) - x)\right)$$

By fixing ζ as a large number, the sigmoid function $\sigma(\cdot)$ outputs 1 if $\hat{F}_{i,j}(u,v) > x$. Basically, $A_{u,v}(x)$ accounts for the number of DCT coefficients at spatial frequency (u,v) that larger than x. After that, $H_{u,v}$ is obtained by differentiating neighboring bins of $A_{u,v}$:

$$H_{u,v}(x) = A_{u,v}(x) - A_{u,v}(x+1),$$

which can be done efficiently by 1D convolution with kernel $[1,-1]$.

To implement histogram layer, we need to select a number of hyperparameters:

- Choosing x, ζ: In [129], x is treated as variables, initialized as 101 integers from -50 to 50 (typical range of DCT coefficients), and tunable during training. On the other hand, ζ is fixed to 10^6. This setting leads to vanishing gradients in our problem since we deal with integer DCT coefficients while [129] deals with real numbers. Therefore, the sigmoid function always reaches its extremes, making the update of

x infeasible. Therefore, we finally fix x as 101 constant bins from -50 to 50 for the efficiency.

- Choosing $\mathfrak{B}$: We use $\mathfrak{B} = 64$, similarly to the patch size used in [56].
- Choosing spatial frequencies: We use all AC frequencies on luminance channel, and just omit DC frequency.

The output of histogram layer is 63×101 (63 AC frequencies, and 101 bins) are consecutively fed to 3 consecutive convolutional layers of receptive field 5×5 with kernels dimension $8, 16, 32$, respectively. Then, tanh activation is added after each convolution layer to obtain nonlinearity, followed by max pooling with stride $[2, 2]$ to reduce the size of feature maps. We, afterwards, incorporate the statistical moments layer described in [131] to extract 4 statistical values (max, min, mean and variance) of each of 32 feature maps. This layer significantly reduces dimensionality of feature maps, and stabilizes the training. Since variance is computed based on realizations and mean, end-to-end training with both variance and mean computation might lead to instabilities, thus we treat variance as a constant in P-CNN. Subsequently, max, min and mean values of each of feature maps are concatenated in a vector of 96 features (3×32) and fed to Multilayer Perceptron (MLP) classifier (two hidden layers of $128, 64$ hidden units and one output layer of $\mathcal{C}$ output units). Probabilities over $\mathcal{C}$ classes are computed through $\mathcal{C}$-way softmax, obtaining P-CNN predictions. P-CNN is trained up to 100 epochs using Adam optimizer. The initial learning rate is set to 10^{-4} and exponentially decayed after 50 epochs to improve convergence. Batch normalization, dropout, and weight regularization are *not* applied during training.

4.4.2 Combination with Metadata-Based Features

As SNs employ their own JPEG compression and resizing, quantization tables and image size can be considered as useful cues. Furthermore, specifications of encoding process also present peculiarities of SN platforms. In this study, we explore the ability to combine deep features from P-CNN, called PCnnFea, and MetaFea.

PCnnFea consists of 4 statistical values (max, min, mean, variance) of each of 32 feature maps output from the third convolutional layer of P-CNN. Its dimensionality is $4 \times 32 = 128$, as shown in Figure 4.5. Although variance is treated as constant during P-CNN training due to instability, once P-CNN is optimal, these features encode the variability of feature maps. Thus, final classifier named as P-CNN-FF (P-CNN with Feature Fusion), exploiting both types of features, use 128 deep features (4×32) concatenated with 152 metadata-based features, thus working on a 280-d feature vectors. P-CNN-FF is

simply a Multi-Layer Perceptron (MLP) classifier with two hidden layers of $128, 64$ hidden units and an output layer of $\mathcal{C}$ output units. P-CNN-FF is trained for 100 epochs using Stochastic Gradient Descent (SGD) with momentum 0.9, initial learning rate 10^{-2}. The learning rate is divided by 5 if the training loss does not decrease after two consecutive epochs.

4.5 Experiments

In this section, extensive tests are carried out to evaluate the goodness each type of features, the capability of classification methods under different test scenarios. We first introduce in Section 4.5.1 benchmarking datasets used in our experiments, and in Section 4.5.2 general test settings. The remaining three sections concentrate on results and discussions.

4.5.1 Benchmarking datasets

A. Images Sharing via Instant Messaging App (ISIMA)

As the name suggest, this dataset contains images shared via instant messaging apps. We first select images from different mobile devices to have variety of resolution, JPEG compression quality and attached metadata. To do this, images are taken from Vision [90] dataset, a recently proposed dataset for source identification, that contains JPEG images captured from 35 different mobile devices. 10 images are taken under default settings on each camera, thus ending up with 350 original images. Original images are then manually shared via Facebook Messenger (M), Telegram (T) and WhatsApp (W), by using both an Android system and an iOS system.

Since the processing of images is reasonably performed in the uploading phase, so the focus is shifted on the sending OS (Android to iOS and iOS to Android instead of four possible combinations of Android and iOS). The single sharing schema is described in Figure 4.6. By doing so, the total number of collected images is $350 \times 3 \times 2 = 2100$.

After a first sharing, it is common that images are shared once again either via the same app or through another app, thus leading to multiple shared images. In order to study this scenario, the set of images used in the previous section is extended by adding images that are shared twice. In particular, each of the 350 original images is first shared through one of the apps and then re-shared through another app, among the three ones considered. In order to study the interaction among OSs but at the same time avoiding the dataset size to explode, the double sharing is performed successively either by iOS-Android or by Android-iOS. The double sharing scenario considered is depicted in Figure

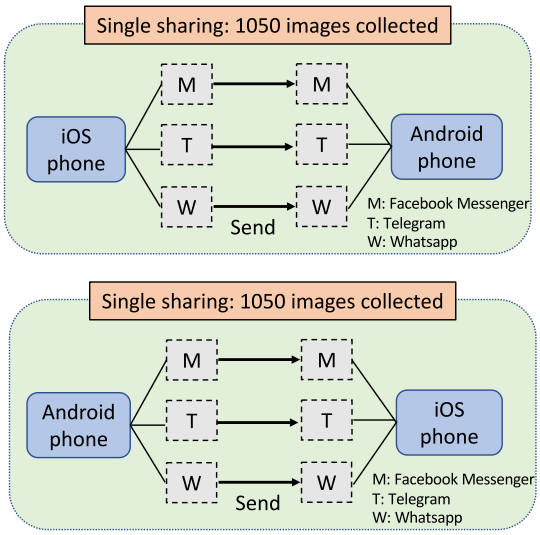


Figure 4.6: Single sharing scenario in ISIMA.

4.7. By doing so, $3 \times 3 \times 2 = 18$ different app/OS combinations lead to $350 \times 18 = 6300$ double shared images, plus the 350 original images.

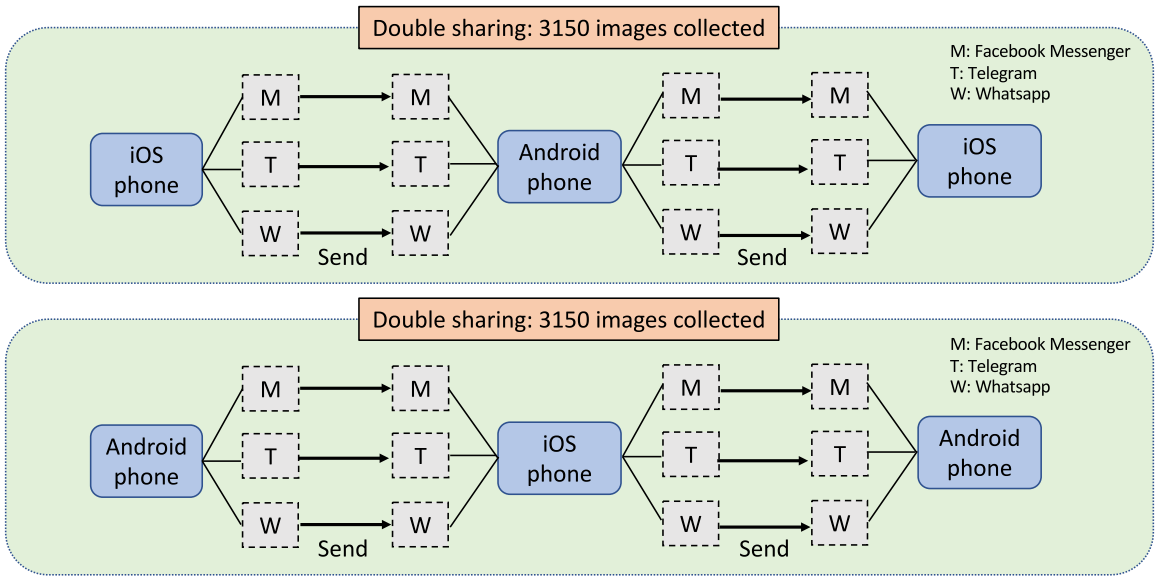


Figure 4.7: Double sharing scenario in ISIMA.

In the end, ISIMA dataset contains $350 + 2100 + 6300 = 8750$ images. We made this dataset available at `http://loki.disi.unitn.it/ISIMA/`.

B. RAISE Social Multiple Up-Download (R-SMUD)

To study in-depth multiple sharing scenario, we create a large-scale dataset of images that are uploaded over multiple SNs. R-SMUD contains images shared over three SNs: Facebook (FB), Flickr (FL) and Twitter (TW). 50 images (raw format) have been selected from RAISE and cropped on top-left corner at sizes: 377×600, 1012×1800 and 1687×3000 with an aspect ratio of $9:16$. All cropped images are subsequently JPEG compressed (the independent JPEG group's software has been adopted) at quality factors $QF = \{50, 60, 70, 80, 90, 100\}$. This yields to 900 images in total, shared at maximum 3 times through the three platforms. By considering all the possible combinations with repetitions we have totally $\mathcal{C} = \sum_{k=1}^{J} (SN)^k$ classes, where SN represents the number of social networks ($SN = 3$) and J indicates the maximum number of sharing (e.g., for $J = 3$ it yields $\mathcal{C} = 3^1 + 3^2 + 3^3 = 39$).

To the end, this dataset consists of $900 + 39 \times 900 = 36,000$ images. Technically, we have developed *bots* to support automatic upload/download. This dataset can be downloaded from: `http://loki.disi.unitn.it/~rvsmud`.

C. Vision Social Multiple Up-Download (V-SMUD)

The previous dataset, R-SMUD, originates from raw images. Differently, we develop V-SMUD dataset by selecting 510 original JPEG images from Vision dataset [90] (15 for each of the 34 cameras, except camera number twelve *Sony XperiaZ1 Compact* that it is excluded since all of its images exceed 5 MBs, which is the image size limit by Twitter [1]). All selected images are then shared via FB, FL, TW, at maximum 3 times.

In the end, V-SMUD consists of $510 + 39 \times 510 = 20,400$ images. This dataset can be downloaded from: `http://loki.disi.unitn.it/~rvsmud`.

D. Ucid Social (U-SOC)

U-SOC dataset [2] has been proposed by [54]. 1338 raw images (512×384 pixels) of UCID dataset [132] are JPEG compressed with $QF = \{50, 55, 60, 65, 70, 75, 80, 85, 90, 95\}$. For each of QF, 1000 images are uploaded/downloaded on FB, FL, TW, resulting $10 \times 1000 \times 3 = 30,000$ images in total. For multiple sharing scenario, images uploaded the first time are subsequently downloaded and cross-uploaded the second time via another SN, resulting totally $60,000$ images.

In the end, U-SOC consists of $10,000 + 10,000 \times 9 = 100,000$ images.

[1] https://developer.twitter.com/en/docs/media/upload-media/uploading-media/media-best-practices.html, last access: 25/10/2018

[2] http://lci.micc.unifi.it/labd/2015/01/trustworthiness-and-social-forensic/, last access: 04/01/2019

4.5.2 Settings

We evaluate different methodologies:

- Caldelli et al. [54]: this method relies on SignalFea and RF classifier.
- Amerini et al. [56]: this method relies on CNNs trained on pre-extracted histogram of DCT coefficients. [56] provides predictions on patch (64×64) level, the predictions on image level is taken by majority voting.
- LR-HCF, SVM-HCF, RF-HCF: these three methods rely on HanCrFea with three classical classifiers LR, SVM, RF, respectively.
- P-CNN: as mentioned in Section 4.4.1, this method relies on CNNs with histogram layer. Therefore, it can be trained directly on DCT coefficients. P-CNN provides predictions on patch (64×64) level, the predictions on image level are done by majority voting.
- P-CNN-FF: as mentioned in Section 4.4.2, this method relies on the combination (concatenation) of PCnnFea and MetaFea, and a MLP classifier. P-CNN-FF provides predictions on patch (64×64) level, the predictions on image level are done by majority voting.

All classical classifiers are trained and tested following one-vs-all strategy, regarding to fitting a single classifier to distinguish a single class against the rest.

We use two protocols for the evaluation:

- Strategic cross-validation [54]: From the dataset analyzed in each case, 30% of images are used for training and 70% for testing. We circularly shift the training set by 5 images until it covers the whole dataset and reporting the average accuracy.
- Pre-splitting [56]: As strategic cross-validation is not suitable to CNN-based approaches due to multiple splittings of training and test set, we follow [56] and split the each dataset into training, validation and test set with the proportion 80%, 10% and 10%, respectively. The performance is obviously measured on the test set in terms of accuracy.

4.5.3 Modifications performed by SNs

In this section, we analyze the operations of SNs on our two large-scale collected datasets, namely R-SMUD and V-SMUD. In particular, we analyze whether or not the uploaded image is compressed, the corresponding compression quality (QF), and the maximal allowable dimension of the image (Max dim.) in single sharing scenario. Note that, the QF

can only be retrieved exactly if the quantization table (only of luminance channel for the sake of brevity) is the scaled instance of the standard quantization table [53]. Otherwise, the QF computed is a real-valued number rather than integer. Details of the analysis

	Compress	QF	Max. dim	Extra info.
FB	Yes	Fine-grained	2048	A few images are pixel-wise unmodified
TW	Yes	Original QF, 85	1200	A few images are pixel-wise unmodified
FL	Yes	Fine-grained	2048	-

Table 4.1: Operations performed by SNs on R-SMUD.

on R-SMUD is shown in Table. 4.1. We can observe that all SNs re-compress uploaded images. FB and FL use fine-grained QF rather than a fixed one, while TW occasionally reuses the same original QF, and 85. All SNs limit the maximum dimension of an image. It can be also noted that a few images are pixel-wise unmodified by FB and TW even if they are re-compressed.

The same operational behaviors have been confirmed on V-SMUD, see Table. 4.2, except that all images have been modified after being uploaded. Obviously, as original images selected from Vision are subject to default JPEG compression with high QF which in theory can be further effectively re-recompressed. On the contrary, some of original images from of R-SMUD have been aggressively compressed, i.e. $QF = 50$. In multiple

	Compress	QF	Max. dim	Extra info.
FB	Yes	Fine-grained	2048	-
TW	Yes	85	1200	-
FL	Yes	Fine-grained	2048	-

Table 4.2: Operations performed by SNs on V-SMUD.

sharing scenario, the number of possible sharing paths increase exponentially. To examine operations applied at each step, we pick an original image from R-SMUD and V-SMUD and report the image dimensions (maximal dimension $\times$ minimal dimension) as well as QF in Table 4.3.

For IMAs, the modification policies are further complex. They depend not only on apps, but also on the interplay of device OS and screen resolution. In general, the images are resized and re-compressed after being shared via IMAs. In the next section, we first analyze the SN origin of digital images with respect to instant messaging apps.

4.5.4 Analysis on Instant Messaging Apps

In the context of instant messaging apps, we are interested in understanding the interplay of the apps. Following strategic cross-validation, we first evaluate LR-HCF, SVM-HCF,

R-SMUD		V-SMUD	
SN chain	Image dimensions, QF	SN chain	Image dimensions, QF
FB	$2048 \times 1151, 90$	FB	$2048 \times 1536, 77$
TW	$1200 \times 675, 85$	TW	$1200 \times 900, 85$
FL	$2048 \times 1152, 85$	FL	$2048 \times 1536, 81$
FB-FB	$2048 \times 1151, 90$	FB-FB	$2048 \times 1536, 77$
FB-TW	$1199 \times 674, 85$	FB-TW	$1200 \times 900, 85$
FB-FL	$2048 \times 1151, 87$	FB-FL	$2048 \times 1536, 74$
TW-FB	$1200 \times 675, 71$	TW-FB	$1200 \times 900, 71$
TW-TW	$1200 \times 675, 85$	TW-TW	$1200 \times 900, 85$
TW-FL	$1024 \times 576, 90$	TW-FL	$1024 \times 768, 83$
FL-FB	$2048 \times 1151, 77$	FL-FB	$2048 \times 1536, 71$
FL-TW	$1200 \times 675, 85$	FL-TW	$1200 \times 900, 85$
FL-FL	$2048 \times 1152, 82$	FL-FL	$2048 \times 1536, 77$
FB-FB-FB	$2048 \times 1151, 90$	FB-FB-FB	$2048 \times 1536, 77$
FB-FB-TW	$1199 \times 674, 85$	FB-FB-TW	$1200 \times 900, 85$
FB-FB-FL	$2048 \times 1151, 87$	FB-FB-FL	$2048 \times 1536, 74$
FB-TW-FB	$1199 \times 674, 71$	FB-TW-FB	$1200 \times 900, 71$
FB-TW-TW	$1199 \times 674, 85$	FB-TW-TW	$1200 \times 900, 85$
FB-TW-FL	$1024 \times 576, 91$	FB-TW-FL	$1024 \times 768, 81$
FB-FL-FB	$2048 \times 1151, 88$	FB-FL-FB	$2048 \times 1536, 71$
FB-FL-TW	$1199 \times 674, 85$	FB-FL-TW	$1200 \times 900, 85$
FB-FL-FL	$2048 \times 1151, 85$	FB-FL-FL	$2048 \times 1536, 69$
TW-FB-FB	$1200 \times 675, 71$	TW-FB-FB	$1200 \times 900, 71$
TW-FB-TW	$1200 \times 675, 71$	TW-FB-TW	$1200 \times 900, 71$
TW-FB-FL	$1024 \times 576, 88$	TW-FB-FL	$1024 \times 768, 81$
TW-TW-FB	$1200 \times 675, 71$	TW-TW-FB	$1200 \times 900, 71$
TW-TW-TW	$1200 \times 675, 85$	TW-TW-TW	$1200 \times 900, 85$
TW-TW-FL	$1024 \times 576, 90$	TW-TW-FL	$1024 \times 768, 83$
TW-FL-FB	$1024 \times 576, 83$	TW-FL-FB	$1024 \times 768, 71$
TW-FL-TW	$1024 \times 576, 85$	TW-FL-TW	$1024 \times 768, 85$
TW-FL-FL	$1024 \times 576, 87$	TW-FL-FL	$1024 \times 768, 79$
FL-FB-FB	$2048 \times 1152, 77$	FL-FB-FB	$2048 \times 1536, 71$
FL-FB-TW	$1200 \times 675, 85$	FL-FB-TW	$1200 \times 900, 85$
FL-FB-FL	$2048 \times 1152, 74$	FL-FB-FL	$2048 \times 1536, 83$
FL-TW-FB	$1200 \times 675, 73$	FL-TW-FB	$1200 \times 900, 71$
FL-TW-TW	$1200 \times 675, 85$	FL-TW-TW	$1200 \times 900, 85$
FL-TW-FL	$1024 \times 576, 91$	FL-TW-FL	$1024 \times 768, 85$
FL-FL-FB	$2048 \times 1152, 71$	FL-FL-FB	$2048 \times 1536, 71$
FL-FL-TW	$1200 \times 675, 85$	FL-FL-TW	$1200 \times 900, 85$
FL-FL-FL	$2048 \times 1152, 78$	FL-FL-FL	$2048 \times 1536, 74$

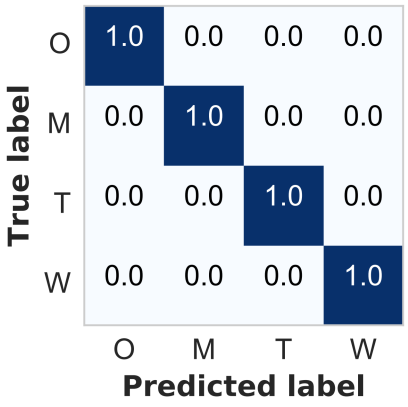
Table 4.3: The properties of image at every SN step. An image of $QF = 80$ and 1687×3000 is selected from R-SMUD and an image of $QF = 99$ and 2560×1920 is selected from V-SMUD.

RF-HCF on two tasks defined on ISIMA dataset:

- APP: identifying whether an image comes from one of the apps, if so, determining the correct one (4 classes, the one for the case of no uploading indicated as O.
- APP+OS: identifying whether an image comes from one of app+OS combinations considered and, if so, determining the correct one (7 classes, including O).

The confusion matrices for these classification tasks are reported in Figures 4.8 and 4.9, together with the average accuracy for all the classifiers. The right figure presents importance scores of the top 10 important features. The color of the bins indicates the kind of feature, according to the legend. It is worth to emphasize that the feature importance scores are taken from RF-HCF.

Confusion matrix



Feature importance

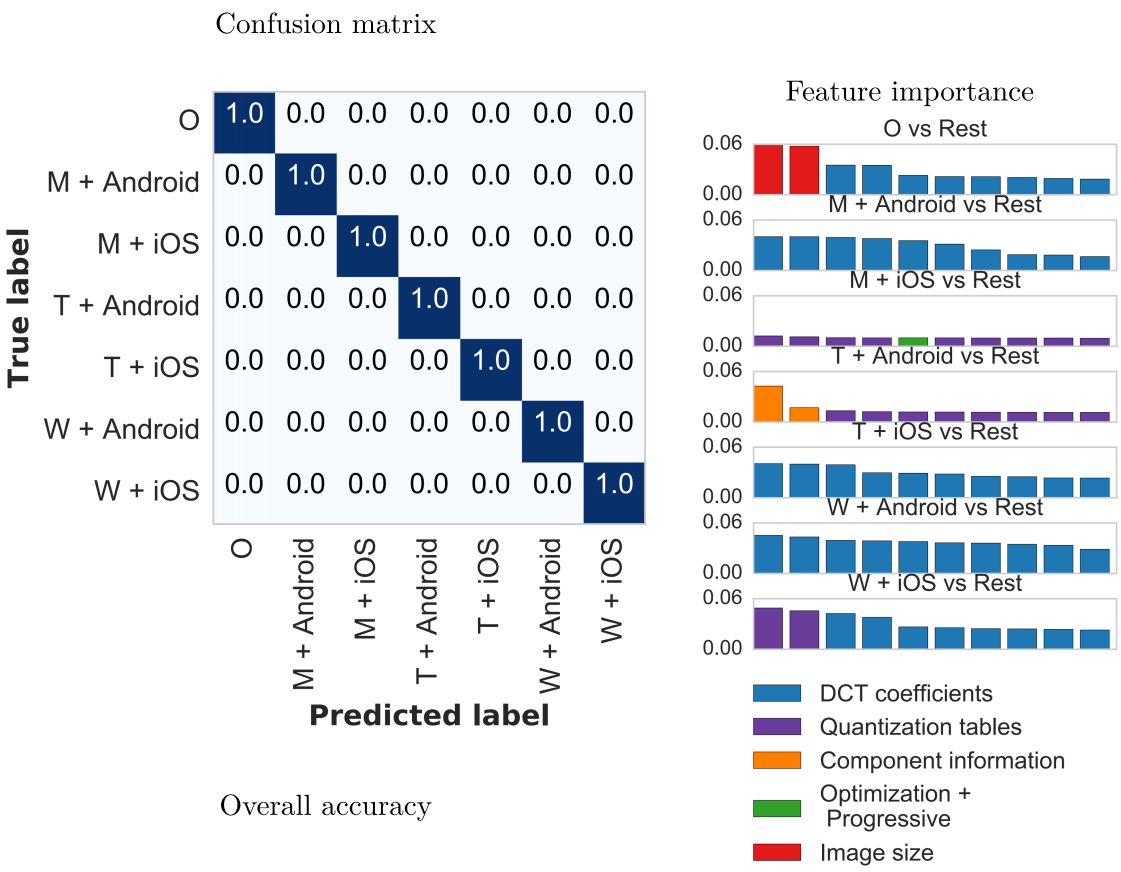
Overall accuracy

LR-HCF	SVM-HCF	RF-HCF
100 ± 0	100 ± 0	99 ± 1

Figure 4.8: Results for the APP task. Better viewed in color.

It turns out that all SN platforms are correctly identified. Interestingly, the ranking of features allows to interpret which kind of features contribute significantly to the results in one-vs-all fashion. As we can see, the image size is critical clue helping to separate original images (never shared) and shared ones. In fact, all apps limit the maximum resolution of images. On the other hand, the apps can be identified based on the distribution of DCT coefficients.

In the second task, APP+OS, we ask whether or not the same app installed on different OSs leave distinctive traces. Perfect classification results on Figure 4.9 answer this question convincingly, that we can identify not only the app but also the OS of the device used to share the image. In order to jointly identify the OS the component information and quantization table values also play a key role, see Figure 4.9.



Overall accuracy

LR-HCF	SVM-HCF	RF-HCF
100 ± 0	100 ± 0	99 ± 2

Figure 4.9: Results for the APP+OS task. Better viewed in color.

Finally, we compare the performance of all methodologies in identifying apps in single sharing scenario of ISIMA. The comparison follows pre-splitting protocol and on image level. Results shown in Figure 4.4 highlight the fact that all methods can identify accurately the apps, particularly methods based on hand-crafted features.

Table 4.4: Results of all methodologies in identifying apps in single sharing scenario of ISIMA.

Method	LR-HCF	SVM-HCF	RF-HCF	Caldelli et al.	Amerini et al.	P-CNN	P-CNN-FF
Accuracy (%)	100	100	100	100	100	98.57	100

4.5.5 Analysis on Public Social Networks

In the context of PSNs, we test all methodologies on the remaining three datasets, namely U-SOC, R-SMUD, and V-SMUD. We decide to use pre-splitting protocol since the dataset size is large. In Table 4.5, performance on image level in single sharing scenario is recorded and presented.

Table 4.5: The accuracy (%) of all methodologies in identifying PSN origin in single sharing scenario.

Method → Dataset ↓	LR-HCF	SVM-HCF	RF-HCF	Caldelli et al.	Amerini et al.	P-CNN	P-CNN-FF
U-SOC	**100**	**100**	**100**	99.31	99.24	92.74	**100**
R-SMUD	**99.26**	98.89	**99.26**	93.70	93.25	89.63	**99.26**
V-SMUD	**100**	**100**	**100**	90.20	98.69	**100**	**100**

Noticeably, LR-HCF, SVM-HCF, RF-HCF and P-CNN-FF achieve perfect performance on U-SOC, while remaining methods are accurate, yet slightly inferior. The four outperforming methods share the same common characteristic; it is the utilization of MetaFea. Since U-SOC originally consists of small-size images, the images then preserve their size after uploaded. The distinctiveness of MetaFea therefore rely on other information rather than image size. This is evident that aside from image size, other information extracted from image metadata is extremely important in the identification of SN origin.

The above observation is again confirmed on R-SMUD and V-SMUD when LR-HCF, SVM-HCF, RF-HCF and P-CNN-FF continue to outperform the remaining methods which exploit only DCT coefficients.

As a summary, the tests in Section 4.5.4 and this Section highlight the feasibility to a posteriori identify SN origin of images. Among different cues, MetaFea appears to be very distinctive, but they have been overlooked by Caldelli et al. and Amerini et al.. Nevertheless, whenever the image is shared/uploaded more than once, the old metadata is overwritten by the new one; thus in principle cannot be useful to identify the first SN platform. In the next section, we analyze in-depth the multiple sharing scenario.

4.5.6 Analysis on Multiple Sharing Scenarios

In double sharing scenario, we attempt to identify the chain of apps on which an image has been circulated through, given zero knowledge about the number of sharing phases. Therefore, in ISIMA dataset we might end up totally 12 categories (classes) in: {M, T, W}$\bigcup${{ M, T, W } $\times$ { M, T, W }}. Similarly, on R-SMUD and V-SMUD we also end up 12 categories: {FB, FL, TW}$\bigcup${{FB, FL, TW} $\times$ {FB, FL, TW}}. The only exception happens to U-SOC where the second SN must be different from the first one. Therefore, the double sharing scenario of U-SOC contains 9 categories only.

We again use pre-splitting protocol, and report performance in terms of accuracy in Table 4.6.

Table 4.6: The accuracy (%) of all methodologies in identifying SN origin in double sharing scenario.

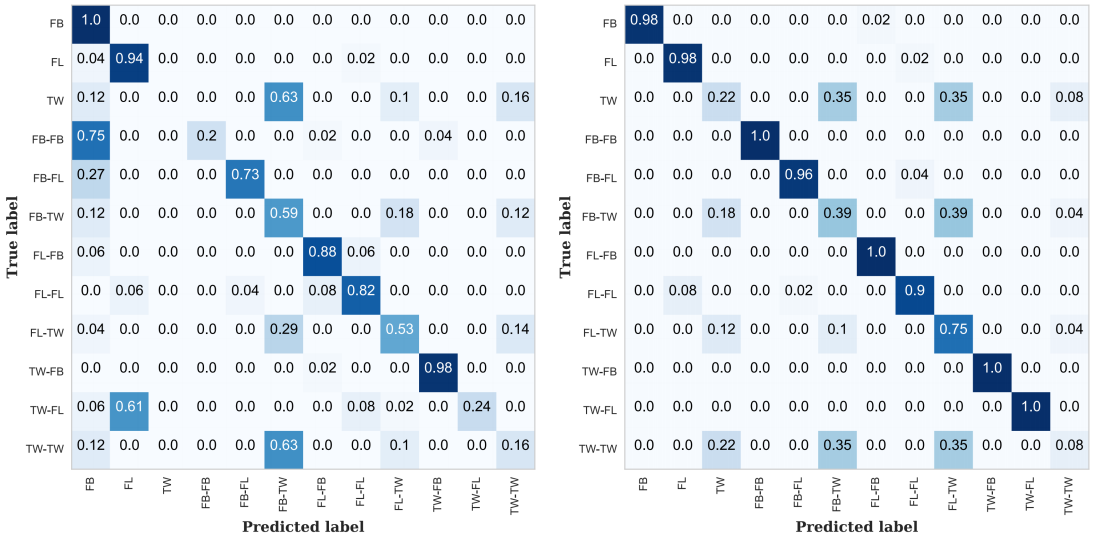
Method → Dataset ↓	LR-HCF	SVM-HCF	RF-HCF	Caldelli et al.	Amerini et al.	P-CNN	P-CNN-FF
ISIMA	61.31	62.26	58.81	50.36	55.71	65.12	**70.60**
U-SOC	69.87	65.83	65.82	50.07	53.71	54.82	**74.10**
R-SMUD	52.22	52.13	59.35	39.91	45.18	43.24	**65.91**
V-SMUD	70.42	71.90	68.30	43.24	54.90	58.82	**77.12**

Without exploiting MetaFea, performance Caldelli et al. and Amerini et al. is modest compared to rest on ISIMA. Exceptionally, P-CNN outperforms traditional classifiers even when only DCT coefficients are used. On the remaining three datasets, we can observe that traditional classifiers with HanCrFea can outperform CNN-based approaches, i.e. Amerini et al. and P-CNN. What makes this is the utilization of MetaFea residing in HanCrFea. In overall, P-CNN-FF achieves state-of-the-art accuracy in double sharing scenario.

To illustrate the superiority of P-CNN-FF, we show the confusion matrices of P-CNN and P-CNN-FF on V-SMUD. Undoubtedly, the diagonal of the confusion matrix in Figure 4.10 (b) is clearer than in Figure 4.10 (a). Zooming closer to Figure 4.10 (b), the misclassifications happen on cases where the last SNs are identical, e.g. TW and FB-TW, TW and FL-TW, TW-TW and FB-TW, TW-TW and FL-TW. This phenomenon is understandable as the same SN platform employs a unique compression and resizing policy, that results in similar peculiarities on the observed image.

We extend the tests to triple sharing scenario on R-SMUD and V-SMUD where images might be shared up to three times, resulting 39 categories. Classical classifiers again outperform CNN-based approaches, see Figure 4.7. Our judgement for this phenomenon is

Figure 4.10: Confusion matrices on V-SMUD dataset in double sharing scenario for (a) P-CNN and (b) P-CNN-FF.



(a) P-CNN (b) P-CNN-FF

two folds: i) MetaFea is useful at least for identifying the last platform, ii) multiple JPEG compressions make the DCT coefficients very sparse and thus CNN-based approaches cannot learn their statistics just from small individual patches. On the other hand, by combining PCnnFea and MetaFea, the best accuracy is achieved by P-CNN-FF.

Table 4.7: The accuracy (%) of all methodologies in identifying SN origin in triple sharing scenario.

Method → Dataset ↓	LR-HCF	SVM-HCF	RF-HCF	Caldelli et al.	Amerini et al.	P-CNN	P-CNN-FF
R-SMUD	25.30	26.21	32.76	17.29	16.95	19.32	**36.18**
V-SMUD	39.57	40.32	40.12	23.68	16.41	27.10	**49.12**

4.6 Conclusions

We have two proposed methodologies for the identification of SN platforms, one based on traditional classifiers with the use of hand-crafted features which are extracted from DCT coefficients and metadata, and another one based on CNNs which can work directly on DCT coefficients. It has been proved experimentally that the metadata-based features are very discriminative, characterizing the JPEG compression of different SNs. Large-scale

datasets are collected to evaluate our methodologies and extensive analyses have been carried out considering: PSNs and IMAs, multiple sharing scenario (up to three times sharing/uploading).

Remaining issues for future investigations include:

- **Stability of DCT coefficients**. Our analyses reveal the fact that it is harder and harder to identify the chain of SNs when an image is circulated multiple times. In fact, the best accuracy in triple sharing scenario is less than 50%. We deem that DCT coefficients are getting *stable* after quantized many times.
- **Complexity of compression and resizing policies**. Different SN platforms employ different policies for storing and transferring images. Unfortunately, such polices are complex and non-stationary. We discover that policies investigated by [55] are too simple. For instance, Facebook stores multiple instances of an image on the server and depending on the device which requests, the appropriate instance will be transferred.

Chapter 5

Source-Target Disambiguation in Copy-Move Forgery

Image copy-move forgery is a process of geometrically transforming an indicated region and applying some means of postprocessing within the same image. Existing algorithms for copy-move detection are able to highlight such nearly duplicated regions but ambiguous in distinguishing source and target.

In this chapter, we propose a four-stream DNN to disambiguate source and target region in copy-move detection. Due to color interpolation and post-processing during copy-move process, the transformation from source to target is irreversible. We show that such a network is able to disambiguate source and target given the forward transformation of the supposed source and the inverse transformation of the supposed target.

Acknowledgement

I would like to thank Prof. Mauro Barni and Dr. Benedetta Tondi from University of Siena for their wise supervision during this study.

Copy-move forgery is one of the most commonly used technique that violates the authenticity of digital images. This operation composes of copying a source region and pasting to a target region in such a way that it is imperceptible. The purpose of copy-move forgery is mainly to conceal an evidence or to deceive viewers. Despite the fact that this is an easy-to-perform operation, detecting it is non-trivial since copy-move forgery is often followed by geometrical transformations such as rotation, resizing, and complex postprocessing. Copy-move detection (hereon we imply the inclusion of localization) methods can be coarsely classified into two categories [133]: i) block-based methods, and ii) keypoint-based methods. Block-based methods split the image into rectangular blocks and extract different kinds of features. On the other hand, keypoint-based methods aim to select high-entropy regions only without splitting the image, and extract local information or features. Both methods afterwards perform matching, and detecting duplicated regions based on extracted features.

This study focuses on another problem after duplicated regions are detected; it is to disambiguate source and target regions. The forensic scenario we considered is illustrated in Figure 5.1. Given the forged image, an effective and efficient copy-move detection method can be used to detect and localize duplicated regions, and after that the forensic investigator is required to indicate source and target region.

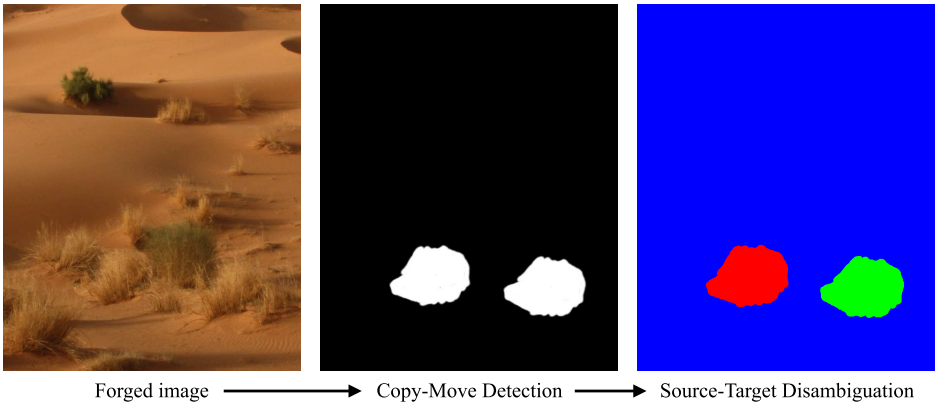


Figure 5.1: Forensic scenario in source-target disambiguation. Left: forged image, middle: duplicated regions, and right: annotation of source and target. Better viewed in color.

5.1 Problem statement

The source region $\mathcal{P}$ is defined on $\{\mathcal{P}_i\}$, where $\mathcal{P}_i = (\mathrm{x}_i^s, \mathrm{y}_i^s)$ is a point on the 2D regular grid. Analogously, the target region $\mathcal{Q}$ is defined on $\{\mathcal{Q}_i\}$, where $\mathcal{Q}_i = (\mathrm{x}_i^t, \mathrm{y}_i^t)$. Copy-move operation can be comprehensible as a geometrical transformation between $\mathcal{P}$ and

$\mathcal{Q}$. Using a transformation $\mathcal{T}_\theta(\cdot)$ parameterized by six parameters, the transformation of a point $\mathcal{P}_i$ in source region results in $\hat{\mathcal{Q}}_i = (\hat{\mathrm{x}}_i^t, \hat{\mathrm{y}}_i^t)$ in target region:

$$\begin{pmatrix} \hat{\mathrm{x}}_i^t \\ \hat{\mathrm{y}}_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \theta_{1,3} \\ \theta_{2,1} & \theta_{2,2} & \theta_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \mathrm{x}_i^s \\ \mathrm{y}_i^s \\ 1 \end{pmatrix}$$

$\mathcal{T}_\theta$ can represent rotation, resizing, sheering, and translation. After transformation, $\hat{\mathcal{Q}}_i$ may deviate from the regular grid points. The pixel value of the forged image I at regular grid point $\mathcal{Q}_i = (\mathrm{x}_i^t, \mathrm{y}_i^t)$, therefore, is interpolated from neighboring pixels of the source region by:

$$I(\mathrm{x}_i^t, \mathrm{y}_i^t) = \sum_{j=1}^{|\mathcal{P}|} I(\mathrm{x}_j^s, \mathrm{y}_j^s) k\left(\mathrm{x}_i^t, \hat{\mathrm{x}}_j^t\right) k\left(\mathrm{y}_i^t, \hat{\mathrm{y}}_j^t\right), \tag{5.1}$$

where $k(\cdot, \cdot)$ is the kernel function drawing spatial relationship of inputs. For instance, if nearest neighbor interpolation is used, Eq. (5.1) becomes:

$$I(\mathrm{x}_i^t, \mathrm{y}_i^t) = \sum_{j=1}^{|\mathcal{P}|} I(\mathrm{x}_j^s, \mathrm{y}_j^s) \delta\left(\lfloor \mathrm{x}_i^t + 0.5 \rfloor - \hat{\mathrm{x}}_j^t\right) \delta\left(\lfloor \mathrm{y}_i^t + 0.5 \rfloor - \hat{\mathrm{y}}_j^t\right),$$

where $\delta(\cdot)$ is the Dirac delta function. With bilinear interpolation, Eq. (5.1) becomes:

$$I(\mathrm{x}_i^t, \mathrm{y}_i^t) = \sum_{j=1}^{|\mathcal{P}|} I(\mathrm{x}_j^s, \mathrm{y}_j^s) \max\left(0, 1 - \left|\mathrm{x}_i^t - \hat{\mathrm{x}}_j^t\right|\right) \max\left(0, 1 - \left|\mathrm{y}_i^t - \hat{\mathrm{y}}_j^t\right|\right). \tag{5.2}$$

Although the transformation is one-to-one mapping, after interpolation $|\mathcal{Q}| \neq |\mathcal{P}|$ as new pixels are produced or duplicated pixels are discarded. Interpolations introduce correlations among neighboring pixels in $\mathcal{Q}$. After interpolation, pixel values in $\mathcal{P}$ and $\mathcal{Q}$ are nearly duplicated. Various postprocessing might be applied globally on the image or locally on the target region in order to hide traces of pasting.

The motivation of this work is based on two observations: i) in realistic copy-move forgery, the composition of interpolation and postprocessing makes the entire copy-move process *non-invertible*, ii) even if copy-move forgery is carried out carefully, subtle artifacts like edge inconsistencies are inevitable. Given the forged image, if one mimics the copy-move process starting from source, i.e. forward copy-move, it is possible to approximate target because it is the correct direction. On the opposite, if one performs copy-move process from target, i.e. backward copy-move, it can hardly approximate source due to non-invertibility.

Based on this reasoning, the disambiguation of source-target can be ideally carried out under two hypotheses:

- $\mathbf{H}_1$: $\mathcal{P}$ is *source*, and $\mathcal{Q}$ is *target*.
- $\mathbf{H}_0$: $\mathcal{Q}$ is *source*, and $\mathcal{P}$ is *target*.

Let us denote $\tilde{\mathcal{Q}}$ the result of forward copy-move made from $\mathcal{P}$, and $\tilde{\mathcal{P}}$ the result of backward copy-move made from $\mathcal{Q}$. Testing the aforementioned hypotheses requires a measure of *approximation* which quantifies how much similar the two regions are. If $\tilde{\mathcal{Q}}$ approximates $\mathcal{Q}$ *better* than $\tilde{\mathcal{P}}$ approximates $\mathcal{P}$, we select $\mathbf{H}_1$. Otherwise, we select $\mathbf{H}_0$. Under the role of a forensic investigator, the complete set of parameters of the copy-move process is unknown. Moreover, the investigator needs a proper metric to assess the approximation of two regions, which is non-trivial for high-dimensional data.

In this work, we propose a solution to estimate parameters of the affine transformation $\mathcal{T}_\theta$, and leverage discriminative power of DNNs for such purpose.

Given complex forms of copy-move in practice, we need to draw a clear boundary of our consideration. In particular, with two hypotheses we consider only single source paired with single target (1-1) as illustrated in Figure 5.2 (a). The case n sources singly paired with n targets in Figure 5.2 (b) can be solved practically as multiple (1-1) cases. More complicated copy-move forgeries that require more than two hypotheses remain for future consideration.

(a) single pairing, 1-1 (b) single pairing, n-n

Figure 5.2: Two forms of copy-move forgeries that can be disambiguated with two hypotheses.

5.2 Siamese Network Approach

We can assume that parameters of postprocessing of copy-move forgery are unknown. Therefore, the simulated copy-move forgeries involve only geometrical transformation and a pre-selected interpolation method, i.e. $\tilde{\mathcal{Q}} = \mathcal{T}_\theta(\mathcal{P})$ and $\tilde{\mathcal{P}} = \mathcal{T}_\theta^{-1}(\mathcal{Q})$, where $\mathcal{T}_\theta^{-1}(\cdot)$ is the inverse transformation. We can verify if $\tilde{\mathcal{Q}}$ approximates well $\mathcal{Q}$ by means of a Siamese network [134] which was originally proposed in 1990s for signature verification and later successfully extended for other computer vision applications. Siamese network

composes of two identical sub-networks, followed by a non-linear classifier that classifies latent representations of two inputs.

For each of M training images, we can extract positive pair $\left(y_1^{(i)} = 1, x_1^{(i)} = \{\mathcal{Q}^{(i)}, \tilde{\mathcal{Q}}^{(i)}\}\right)$, and a negative pair $\left(y_0^{(i)} = 0, x_0^{(i)} = \{\mathcal{P}^{(i)}, \tilde{\mathcal{P}}^{(i)}\}\right)$, where $y_1^{(i)}$ and $y_0^{(i)}$ are positive and negative labels, and $1 \leq i \leq M$. The Siamese network returns logistic prediction $p_j^{(i)}$ on the pair $x_j^{(i)}$ with $j \in [0, 1]$, indicating how probable the pair is correct. It is straightforward to train this network by minimizing binary cross-entropy between logistic predictions and labels. For a test image with $x_1^{(0)} = \{\mathcal{Q}^{(0)}, \tilde{\mathcal{Q}}^{(0)}\}$ and $x_0^{(0)} = \{\mathcal{P}^{(0)}, \tilde{\mathcal{P}}^{(0)}\}$, the hypothesis h is selected by the following rule:

$$h = \begin{cases} 1, & \text{if} \quad p_1^{(0)} > p_0^{(0)} \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

The limitation of such a Siamese architecture is its incapability to jointly learn the dependency of $\{\mathcal{Q}^{(i)}, \tilde{\mathcal{Q}}^{(i)}\}$ and $\{\mathcal{P}^{(i)}, \tilde{\mathcal{P}}^{(i)}\}$. Instead, it treats positive and negative examples independently, while $\{\mathcal{Q}^{(i)}, \tilde{\mathcal{Q}}^{(i)}\}$ and $\{\mathcal{P}^{(i)}, \tilde{\mathcal{P}}^{(i)}\}$ are indeed dependent, i.e. if one is negative, the other one must be positive.

We propose a 4-twin DNN (hereon called 4-Twin Net) as a solution. 4-Twin Net simultaneously receives two pairs as input and outputs conditional probabilities of two hypotheses. Concretely, we train 4-Twin Net on large number of training examples $\left(y_1^{(i)}, x_1^{(i)}, y_0^{(i)}, x_0^{(i)}\right)$. In the next section, we describe in detail the architecture of 4-Twin Net and its training tactics.

5.3 4-Twin Network Approach

5.3.1 Architecture

Our proposed architecture for 4-Twin Net is represented in Figure 5.3. This network composes of four identical stacks of convolutional layers, i.e. all of them share the same weights, and two identical stacks of fully connected layers. The key role of stacked convolutional layers is to finally extract a 512-d feature vector from the $64 \times 64 \times 3$ input image. On the other hand, the feature vector of its counterpart image is extracted in the similar way. To this end, we mention convolutional layers as a feature extractor.

Let us describe the flow of 4-Twin Net when $\left(\mathcal{Q}, \tilde{\mathcal{Q}}, \mathcal{P}, \tilde{\mathcal{P}}\right)$ are fed as inputs. The top two branches extract features of $\mathcal{Q}, \tilde{\mathcal{Q}}$ and combine them via a concatenation operator. The combined feature vector is then passed through an MLP network whose output is pre-normalized score z_1 (or logit). The bottom two branches, on the other hand, receive $\mathcal{P}, \tilde{\mathcal{P}}$ and similarly output z_0 in the end. Due to weight sharing, 4-Twin Net can be

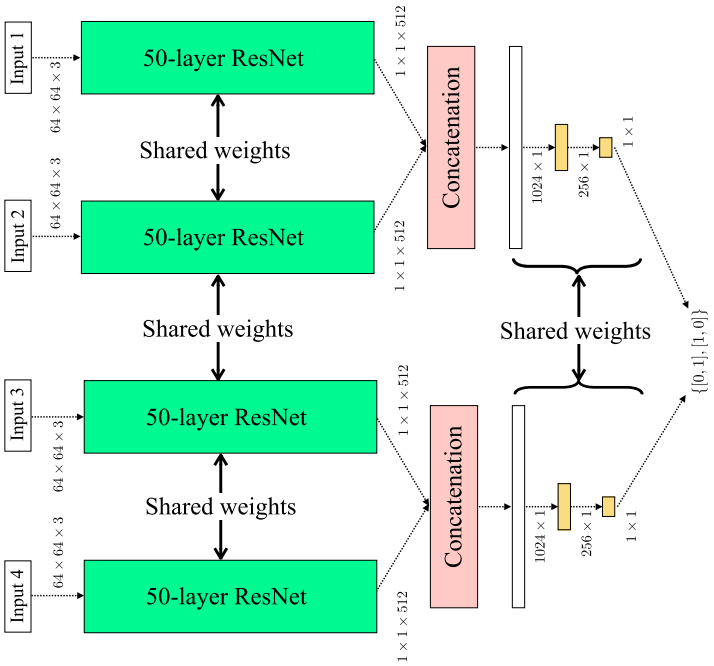


Figure 5.3: The proposed architecture of 4-Twin Net.

interpreted as two standard Siamese networks unless the probabilistic dependency of z_1, z_0 are constructed by the softmax function:

$$p_i = \frac{e^{z_i}}{\sum\limits_{i \in [0,1]} e^{z_i}},$$

where p_i has probabilistic interpretation and $\sum_{i \in [0,1]} p_i = 1$.

Given M training examples $\left\{\left(y_1^{(i)}, x_1^{(i)}, y_0^{(i)}, x_0^{(i)}\right)\right\}_{i \in [1,M]}$, we train 4-Twin Net by minimizing the empirical cross entropy of predictions and input labels written in terms of a loss function L:

$$L = \frac{1}{M} \sum_{j=1}^{M} \sum_{i \in [0,1]} -y_i^{(j)} \log p_i^{(j)}.$$

5.3.2 Motivating The Architecture of 4-Twin Net

The actual architecture of 4-Twin Net is sketched in Figure 5.3. Before coming up to this architecture, we have empirically investigated some other alternatives.

Feature extractor. We have experimented with a simpler architecture of feature extraction net which consists of 4 convolutional layers whose details are shown in Table 5.1. Each row represents details of each convolutional layer. This architecture involves

a total of $\approx$ 25000 parameters including biases. Our preliminary results discourage this architecture since the network does not generalize on our synthetic examples.

Table 5.1: Simple architecture of feature extractor.

Layer name	Input dim	Output dim	Kernel	Max pooling
conv1	$64 \times 64 \times 3$	$32 \times 32 \times 8$	$5 \times 5 \times 8$, stride 1	stride 2
conv2	$32 \times 32 \times 8$	$16 \times 16 \times 16$	$3 \times 3 \times 16$, stride 1	stride 2
conv3	$16 \times 16 \times 16$	$8 \times 8 \times 32$	$3 \times 3 \times 32$, stride 1	stride 2
conv4	$8 \times 8 \times 32$	$4 \times 4 \times 64$	$3 \times 3 \times 64$, stride 1	stride 2

The idea of choosing a deep architecture like 50-layer Residual Network (ResNet) [135] is mainly to ensure that the 4-Twin Net has sufficient capacity, given rich training data. Moreover, such a deep architecture have been successfully employed by [136] in a Siamese network. Details of 50-layer ResNet in our implementation is listed in Table 5.1. It includes totally 50 convolutional layers (conv*). [135] hypothesizes that if stacked non-linear layers can approximate an underlying unknown function, they can also approximate the residual function instead. Therefore, the output of every few layers are added back by their input before passing to next layers. This principle eases the training of very deep architectures. Here for example, the input of conv2_2 block is the summation of the output and input of conv2_1 (after linearly projected to have size consistency).

Feature combination. Before passing through the MLP classifier, we need to decide on how to fuse feature vectors output from convolutional layers. Popular choices can be point-wise absolute difference [137], square Euclidean distance [117], or concatenation [136]. Square Euclidean distance is not encouraged as previously mentioned in [138] that the gradient would vanish once the distance approaches 0, creating a dangerous plateau in landscape of the loss function. Absolute difference function on the other hand is unsmooth (undefined derivative at 0). We choose to implement the concatenation of feature vectors as done in [136]. Concatenation of features, however, breaks the symmetry of 4-Twin Net, i.e. the order of inputs is sensitive. We will introduce a trick during training to mitigate this effect.

MLP. We have tried to concatenate all feature vectors of four branches and come up with one MLP classifier (instead of two). This option doubles the number of parameters in MLP compared with the current use, and exhibits convergence instabilities during training.

Table 5.2: 50-layer ResNet architecture of feature extractor.

Layer name	Input dim	Output dim	Kernel
conv1	$64 \times 64 \times 3$	$32 \times 32 \times 64$	$7 \times 7 \times 64$, stride 2
max_pool	$32 \times 32 \times 64$	$16 \times 16 \times 64$	$3 \times 3 \times 1$, stride 2
conv2_x	$16 \times 16 \times 64$	$16 \times 16 \times 256$	$\begin{bmatrix} 1 \times 1 \times 64 \\ 3 \times 3 \times 64 \\ 1 \times 1 \times 256 \end{bmatrix} \times 3$, stride 1
conv3_x	$16 \times 16 \times 256$	$8 \times 8 \times 512$	$\begin{bmatrix} 1 \times 1 \times 128 \\ 3 \times 3 \times 128 \\ 1 \times 1 \times 512 \end{bmatrix} \times 4$, stride 2
conv4_x	$8 \times 8 \times 512$	$4 \times 4 \times 1024$	$\begin{bmatrix} 1 \times 1 \times 256 \\ 3 \times 3 \times 256 \\ 1 \times 1 \times 1024 \end{bmatrix} \times 6$, stride 2
conv5_x	$4 \times 4 \times 1024$	$2 \times 2 \times 2048$	$\begin{bmatrix} 1 \times 1 \times 512 \\ 3 \times 3 \times 512 \\ 1 \times 1 \times 2048 \end{bmatrix} \times 3$, stride 2
global_avg	$2 \times 2 \times 2024$	$1 \times 1 \times 2048$	-
conv6_x	$1 \times 1 \times 2048$	$1 \times 1 \times 512$	$1 \times 1 \times 512$, stride 1

conv*_x refer to x blocks of convolutional layers. The number of blocks are appended in the details of kernel. max_pool is max pooling layer, and global_avg is global averaging layer.

5.3.3 Training tactics

Training 4-Twin Net requires more particular attentions than training a standard CNN. We present two training tactics which are critical to the success of 4-Twin Net.

Data feeding. Due to feature concatenation before classification, the order of $\mathcal{Q}$ and $\tilde{\mathcal{Q}}$, or of $\mathcal{P}$ and $\tilde{\mathcal{P}}$ is sensitive. However, switching between $\mathcal{Q}$ and $\tilde{\mathcal{Q}}$ as well as between $\mathcal{P}$ and $\tilde{\mathcal{P}}$ should semantically remain the predictions unchanged. To maintain this property, during training we randomly shuffle $\{\mathcal{Q}, \tilde{\mathcal{Q}}\}$ and $\{\mathcal{P}, \tilde{\mathcal{P}}\}$ so that 4-Twin Net does not learn the order. On the opposite, switching between $\{\mathcal{Q}, \tilde{\mathcal{Q}}\}$ and $\{\mathcal{P}, \tilde{\mathcal{P}}\}$ should invert the labels because the target now is positioned as the source, i.e. $[1, 0]$ becomes $[0, 1]$. With this respect, we randomly interchange the role of $\{\mathcal{Q}, \tilde{\mathcal{Q}}\}$ and $\{\mathcal{P}, \tilde{\mathcal{P}}\}$ accompanied with inverted labels.

Batch normalization (BN). A typical pipeline in machine learning applications often involves a feature normalization step which convert features into a fixed distribution, avoiding feature domination and accelerating gradient-based optimizations. Similarly, in our architecture BN [139] is useful for training the convolutional layers. BN normalizes

layer inputs to zero-mean and unit-variance by accumulated means and standard deviations of mini-batches. This procedure is straightforward only with standard CNNs, yet it is not in 4-Twin Net since data flow through four branches. To avoid the accumulated means and standard deviations being biased, we have to ensure within each mini-batch that each of four branches are fed with both four categories, i.e. $\mathcal{Q}, \tilde{\mathcal{Q}}, \mathcal{P}, \tilde{\mathcal{P}}$ by employing the previously described data feeding. More importantly, we accumulate the statistics on one branch only and broadcast to other branches as well in order to make feature extraction part in four streams identical.

Patch cropping. DNNs fix data input size, thus each image region must be preprocessed. We employ a simple strategy: create a bounding box of each region, only resize if one dimension of the bounding box is smaller than 64, and perform centering cropping to 64×64.

5.4 Transformation Estimation

Existing copy-move detection methods provide a (binary) tampering map highlighting source-target regions, yet not all of them provide the estimation of geometrical transformation. As our proposed method requires the transformation matrix, being able to estimate it from the tampering map is essential. This ability is important since there is no perfect detection method that works on every condition. Therefore, it is encouraging to have a method that can be incorporated on top of *any* copy-move detection method. Our goal is to estimate a homography matrix that is used to transform the region $\mathcal{P}$ to

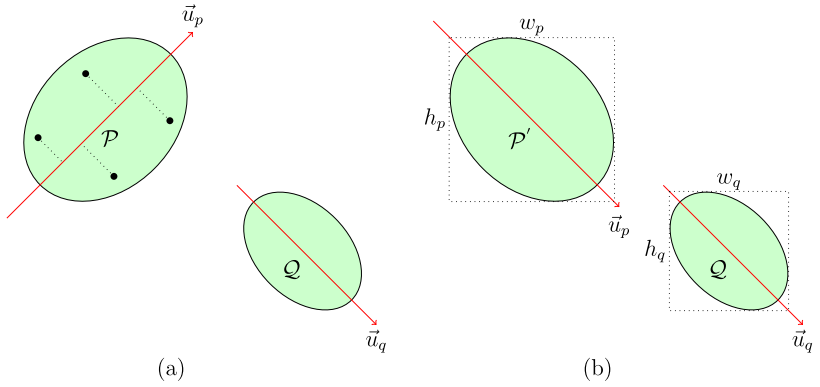


Figure 5.4: Illustrations of rotation angle and resizing factor estimation.

$\mathcal{Q}$ given the binary tampering map. The estimation is done via three steps: i) estimate rotation angle α, ii) estimate resizing factors f_x, f_y, and iii) estimate the translation t_x, t_y.

To estimate α, we find two principle directions of $\mathcal{P}$ and $\mathcal{Q}$ and differentiate their

angles. As real objects are not perfectly circular, the covariance of x-coordinates and y-coordinates therefore is non-zero. Inspired by the idea of Principle Component Analysis (PCA) [140], for each region we attempt to find a direction on which the variance of projected points is maximized. Such principle components are illustrated as u_p, u_q in Figure 5.4 (a). Denote P the $2 \times N$ matrix whose i-th column P_i represents 2D coordinates of point i within $\mathcal{P}$ ($1 \leq i \leq N$); and $\bar{P} = \frac{1}{N}\sum_{i=1}^{N} P_i$ the mean of $\mathcal{P}$. If we assume $u_p^T u_p = 1$, the variance of projected points in $\mathcal{P}$ is given by:

$$\frac{1}{N}\sum_{i=1}^{N}\left(u_p^T P_i - u_p^T \bar{P}\right)\left(u_p^T P_i - u_p^T \bar{P}\right)^T = u_p^T S_p u_p,$$

where $S_p = \frac{1}{N}\sum_{i=1}^{N}\left(P_i - \bar{P}\right)\left(P_i - \bar{P}\right)^T$ is the covariance matrix of ordinary points in $\mathcal{P}$. The principle component u_p is found by:

$$u_p = \arg\max_u u^T S_p u \;\text{ s.t. }\; u^T u = 1.$$

It can be demonstrated that u_p is the eigenvector corresponding to the largest eigenvalue of S_p. We find u_q in a similar way. The angle α is computed as:

$$\alpha = \tan^{-1}\left\{\frac{(u_q)_2}{(u_q)_1}\right\} - \tan^{-1}\left\{\frac{(u_p)_2}{(u_p)_1}\right\}.$$

Afterwards, $\mathcal{P}$ is rotated by the angle α, obtaining $\mathcal{P}'$. By finding the $h_p \times w_p$ bounding box of $\mathcal{P}'$ and the $h_q \times w_q$ bounding box of $\mathcal{Q}$, as illustrated in Figure 5.4 (b), the resizing factors are computed as: $f_x = \frac{w_q}{w_p}, f_y = \frac{h_q}{h_p}$. Finally, the translation is merely the difference of mean of $\mathcal{P}'$ after resized by f_x, f_y and mean of $\mathcal{Q}$.

5.5 Experimental Methodologies

5.5.1 Synthetic Dataset

Training 4-Twin Net requires large amount of data. [80] proposes USCISI, a synthetic dataset of $100,000$ images with annotated source-target, where $80,000$ images can be used for training. Nevertheless, training with only $80,000$ images is insufficient and prone to overfitting to specific settings of that particular dataset. To this end, we develop a large-scale synthetic dataset of $800,000$ images with diverged settings, named as SYN. The development of SYN involves four steps, as shown in Figure 5.5.

Background preparation. We first create a pool of images from RAISE_2k [111], Dresden [3] and Vision [90] datasets. This set includes approximately $28,000$ images (both raw and JPEG formats). For each host image we generate multiple forged instances by

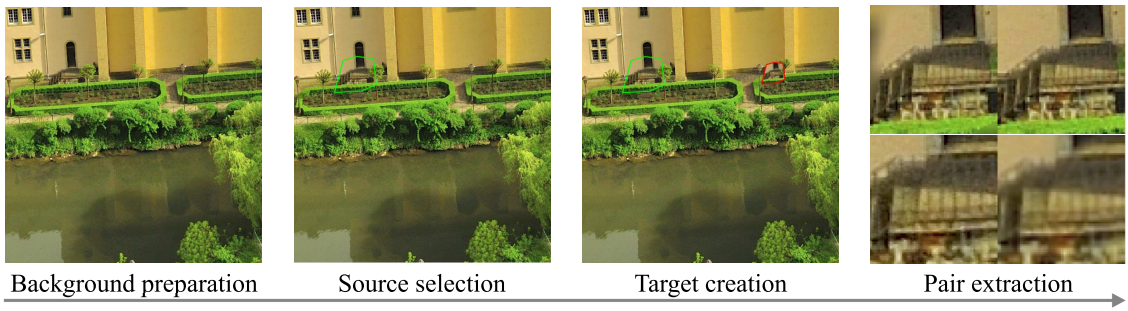


Figure 5.5: Four steps in synthetic dataset (SYN) creation.

working on random cropped portions whose crop size is 1024×1024. Images having minimum dimension smaller than 1024 are skipped.

Source selection. The 1024×1024 image is split into four quadrants, the source is generated from one of them. We generate random convex polygons whose vertices always stay within bounding boxes of $256 \times 256, 128 \times 128, 64 \times 64$ randomly positioned within the selected quadrant. Pixels inside the convex polygon are selected as source.

Target creation. The target is created via an affine transformation of the source. We consider rotation, resizing, resizing after rotation, and rotation after resizing. Rotation angles are randomly picked in $[2, 180]$, step 2, while horizontal and vertical resizing factors are randomly picked in $[0.5, 2.0]$, step 0.01. We use bilinear interpolation as described in Eq. (5.2). Translation is done at last to move the transform of source to one of three remaining quadrants. To improve the forgery quality, we blur the boundary of the target as follows: detecting edge from the target mask by using a highpass filter 5×5, performing binary dilation for 5 iterations to emphasize the edge, and applying averaging filter whose size is randomly selected from $\{3 \times 3, 5 \times 5, 7 \times 7, 9 \times 9, 11 \times 11\}$ on the edge. Finally, we simulate various postprocessing globally on the forged image. Each processing is accompanied with a selection probability as detailed in Table 5.3. 50% of images are unprocessed (probability 0.5), and another 50% are processed under different operations and parameters.

Pair extraction. Given the forged image I with *assumed* source $\mathcal{P}$ and target $\mathcal{Q}$, the transformation $\mathcal{T}_\theta$, we proceed to extract the transform of source $\tilde{\mathcal{Q}} = \mathcal{T}_\theta(\mathcal{P})$ and the inverse transform of target $\tilde{\mathcal{P}} = \mathcal{T}_\theta^{-1}(\mathcal{Q})$. θ is known from the previous step. Since 4-Twin Net disallows arbitrary shapes of inputs, we decide the fit a rectangular bounding box to every of four regions. Let denote the bounding box of $\tilde{\mathcal{Q}}$ as $\tilde{\mathcal{Q}}^b = \left\{\tilde{\mathcal{Q}}_i^b\right\}$. The pixel value at coordinates $\tilde{\mathcal{Q}}_i^b$ is set to the pixel value at $\mathcal{T}_\theta^{-1}(\tilde{\mathcal{Q}}_i^b)$ which belongs to the neighborhood of $\mathcal{P}$. Clearly, if $\mathcal{T}_\theta^{-1}(\tilde{\mathcal{Q}}_i^b)$ deviates from the regular grid, it is interpolated from its neighborhood. The key difference of this step and the copy-move forgery before is

Figure 5.6: Two 64×64 pairs extracted from the forged image.

that the values of pixels within $\tilde{\mathcal{Q}}^b$, but outside $\tilde{\mathcal{Q}}$ are interpolated from the neighborhood of source. The similar procedure applies to $\tilde{\mathcal{P}}$. Finally, we resize each region such that the minimum dimension is 64 and crop the remaining dimension to 64 as well. Examples of four regions are presented in Figure 5.6. As our synthetic images satisfy the required size of 4-Twin Net, no further preprocessing is needed for training and test.

Table 5.3: Postprocessing applied with selected probabilites.

Processing	Matlab command	Prob.
Identity	–	0.5
Lowpass filter	`fspecial('gaussian',3,.5)`	0.017
	`fspecial('gaussian',3,1.)`	0.017
	`fspecial('gaussian',3,1.5)`	0.017
	`fspecial('gaussian',3,2.)`	0.017
	`fspecial('average',3)`	0.017
Highpass filter	`fspecial('unsharp',.5)`	0.017
Denoising filter	`wiener2(·,[3,3])`	0.05
	`wiener2(·,[5,5])`	0.05
Noise adding	`imnoise(·,'gaussian',0,.001)`	0.1
Tonal adjustment	`imadjust(·, stretchlim(x,2/100),[],.8)`	0.033
	`imadjust(·, stretchlim(x,6/100),[],.8)`	0.033
Histogram equalization	`histeq(·)`	0.033
JPEG compression	`qfs = 55:5:100` `imwrite(·, ·, 'Quality',randi(length(qfs)))`	0.1

5.5.2 Settings

Evaluation metric. Given one forged image with single source-target, there are two possible directions: source → target, or target → source. A prediction $[1, 0]$ accepts the direction source → target, and at the same time rejects the direction target → source. We simply use accuracy as the evaluation metric, which is the ratio between the number of correct predictions to the number of images. If the forged regions are known (in terms of binary mask), a correct prediction means the source and target are correctly marked.

Benchmarking datasets. We evaluate our model on four datasets with source-target annotated:

- USCISI. Introduced by [80], this dataset consists of $80,000$ training, $10,000$ validation and $10,000$ test images. All background images are taken from SUN2012 dataset [141] and Microsoft COCO [142] that provide object segmentation mask. Objects are copied and moved by means of geometrical transformations.

- CASIA [1] [143]. This is known as the largest public benchmarking dataset for image forgery detection. A subset of 1313 copy-move forged images are manually selected out of 5123 tampered ones by [80]. Afterwards, source and target regions are labeled by differentiating the tampered and the pristine image.

- CoMoFoD [1]. This dataset consists of 200 base images that are forged without postprocessing. Six kinds of postprocessing are afterwards performed globally on 200 base images. Each postprocessing comes with different parameters, ending up 25 categories of postprocessing ($5,000$ images totally). The source-target annotation is done in a similar manner as with CASIA by [80].

- Grip [144]. This dataset includes 80 base images with rigid copy-move. Two geometrical transformations, i.e. rotation and resizing, and two postprocessing, i.e. local noise addition, and global JPEG compression, are performed on top of base images using the software in [133] to generate 41 categories (3280 images totally). Source/target of all forged images are annotated by ourselves, considering the information of top-left coordinates of source-target given by the software as clues.

Test models. Based on two available large-scale datasets, namely SYN and USCISI, we create and evaluate two models:

- 4-Twin Net*. $800,000$ images in SYN are randomly split into training and validation set with splitting ratio 9:1. We train 4-Twin Net for approximately 66 epochs ($375,000$ iterations with batch size 128) using Adam optimizer. The learning rate is

[1] http://forensics.idealtest.org/casiav2

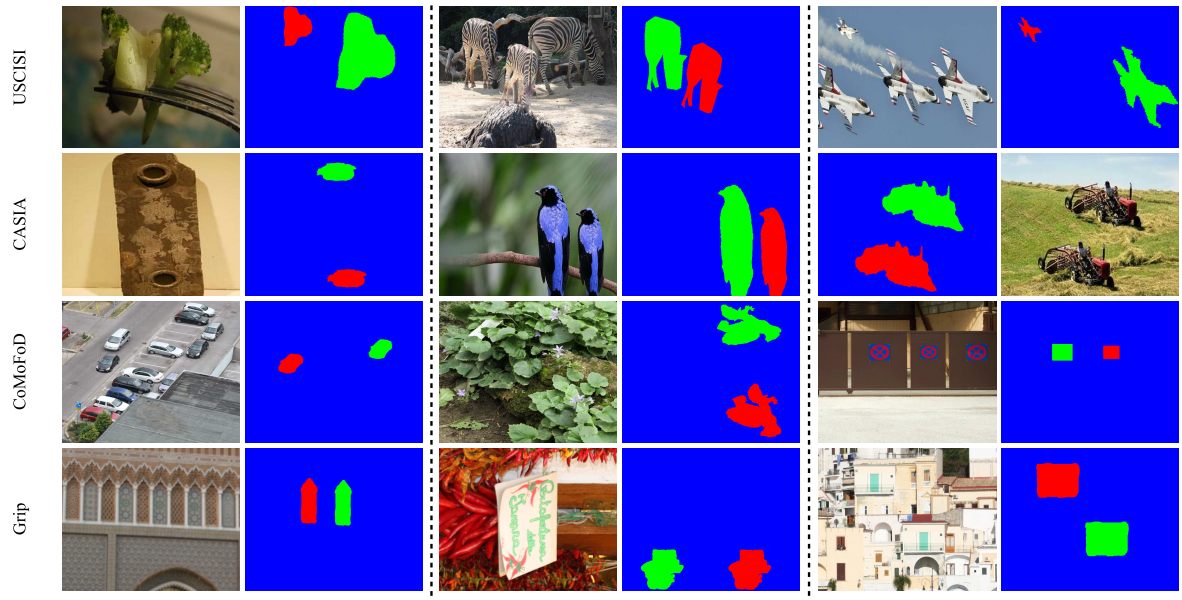


Figure 5.7: Examples of forged images on four datasets. Red: target, green: source, blue: background. All images are resized to 5:4 aspect ratio.

set to 10^{-4}, and after 45 epochs the learning rate is divided by a factor of 2 every 10 epochs.

- 4-Twin Net**. 80,000 training images from USCISI and 150,000 images from SYN are used for fine-tuning 4-Twin Net*. This dataset of 230,000 images is also randomly split into training and validation set with the splitting ratio 9:1. We fine-tune 4-Twin Net* on this new dataset for 150,000 iterations starting from learning rate 5×10^{-5} using Adam optimizer. The learning rate after 25,000 iterations is divided by a factor of 2 every 25,000 iterations.

5.6 Experimental Results

In this section, we present experimental results under different scenarios:

- *Source-target disambiguation given localization and transformation.* We are interested in the discernibility of 4-Twin Net if the two regions (source and target) are localized and the transformation matrix is given. To avoid any confusion here, we note that the given transformation could be either from *actual* source to *actual* target or vice versa.

- *Source-target disambiguation given localization.* Under many circumstances, the actual transformation is unavailable. We estimate the transformation matrix from the

given localization map by using the algorithm described in Section 5.4.

5.6.1 Source-target disambiguation given localization and transformation

The transformation matrix is given only by USCISI. We evaluate 4-Twin Net* and 4-Twin Net** on $10,000$ test images. As our consideration is single source and single target, i.e. (1-1), we perform Connected Component (CC) analysis to automatically verify this condition. Especially, binary erosion and deflation are performed on the given localization mask to remove holes and dots before CC analysis. 1567 images are filtered out since they do not obey (1-1) condition. Table 5.4 presents interesting results. Despite being trained only with SYN where images are less visually plausible, 4-Twin Net* can discern source and target in more realistic images from USCISI, with a high accuracy. Once being fine-tuned jointly on USCISI and a subset of SYN, 4-Twin Net** reaches a nearly perfect accuracy. These results are promising, proving the feasibility of our proposed method. In the next section, we expand the test to more practical contexts.

Table 5.4: Accuracy (%) of 4-Twin Net* and 4-Twin Net** on USCISI. Both localization mask and transformation matrix are given.

Dataset	4-Twin Net*	4-Twin Net**
USCISI (8433/10,000)	92.66	99.32

5.6.2 Source-target disambiguation given localization

In this scenario, the transformation is unavailable. We therefore estimate it from the localization mask. We show in Figure 5.8 how the rotation angle is estimated. The top-left and top-right present the input and its localization map of source and target regions. The bottom-left shows the two principle components of two regions (the red line). By rotating one of the region, we obtain two regions that are well aligned. The estimation of resizing factors and translation distance afterwards become easier. By manually checking, we observe that our algorithm is highly accurate. We do not validate the overall accuracy of this algorithm due to the lack of an evaluation metric. We can compute the product of ground-truth transformation matrix and inverse of our estimated matrix, and expect it to be as close as identity matrix. Nevertheless, this approach may be misleading since we do not know whether our estimate approximates the ground-truth transformation matrix or its inverse. In the previous case, the estimated matrix is very accurate, see Table 5.5.

We first test this scenario in the three datasets, namely USCISI, CASIA and CoMoFoD. Another dataset – Grip, is left out since its base images include only rigid copy-move

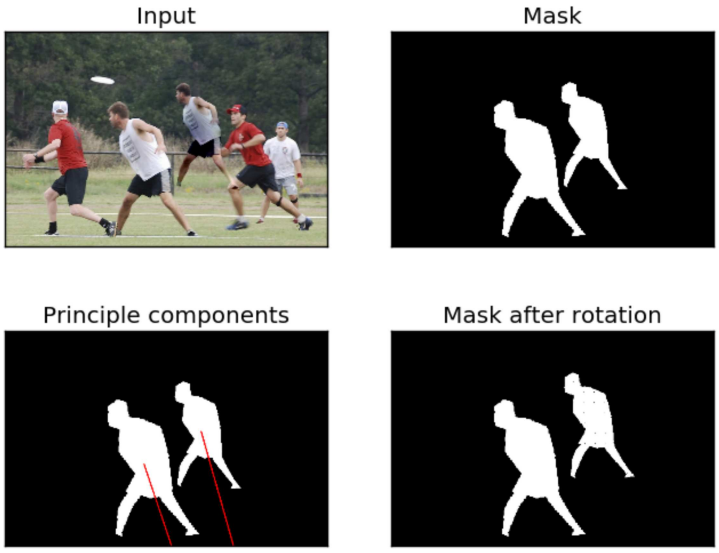


Figure 5.8: The estimation of rotation angle.

Table 5.5: 50-layer ResNet architecture of feature extractor.

Ground-truth transformation matrix	Estimate transformation matrix
$\begin{bmatrix} 1.29 & 0.062 & -238.259 \\ -0.62 & 1.29 & 29.167 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1.293 & 0.064 & -239.482 \\ -0.063 & 1.297 & 29.655 \\ 0 & 0 & 1 \end{bmatrix}$

forgeries, and without postprocessing. By giving a first look at USCISI, we can see that the performance of 4-Twin Net maintain as good as when the transformation matrix is given. This evidence confirms the robustness of our estimation. On two challenging datasets CASIA and CoMoFoD, the number of images satisfying (1-1) condition is 818 and 135, respectively. The performance decreases, as we can see, yet still acceptable. 4-Twin Net** is less accurate than 4-Twin Net* due to the effect of fine-tuning. Our conjecture about the cause of this phenomenon is the diversity difference of SYN and USCISI. SYN includes more diverged rotation angles and rich postprocessing compared to USCISI.

We also validate the discernibility of 4-Twin Net* and 4-Twin Net** under various attacks. With this respect, CoMoFoD provides forged images with different postprocessing including brightness change (BC), contrast adjustment (CA), image blurring (IB), JPEG compression (JC), color reduction (CR), noise addition (NA). Each postprocessing comes with different parameters, see Table 5.7 for more details. Figure 5.9 represents the performance of 4-Twin Net* and 4-Twin Net** under different postprocessing. Among those, image blurring and noise addition and JPEG compression with low quality factors have

Table 5.6: Accuracy (%) of 4-Twin Net* and 4-Twin Net** on three datasets. Only the localization mask is given.

Dataset	4-Twin Net*	4-Twin Net**
USCISI (8433/10,000)	92.93	99.44
CASIA (818/1313)	73.96	71.27
CoMoFoD (135/200)	73.33	71.85

Table 5.7: Naming convention in CoMoFoD [1]. Increasing numbers in [·] of the first column correspond to parameters in the second column.

Naming	Parameters
JC[1-9]	factor = [20,30,40,50,60,70,80,90,100]
NA[1-3]	mean = 0, variance = $[0.009, 0.005, 0.0005]$
IB[1-3]	averaging filter = $[3 \times 3, 5 \times 5, 7 \times 7]$
BC[1-3]	(lower bound, upper bound) = $[(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)]$
CR[1-3]	intensity levels per each color channel = $[32, 64, 128]$
CA[1-3]	(lower bound, upper bound) = $[(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)]$

strong impact, significantly degrading the performance. We note that, the size of forged regions in CoMoFod dataset is particularly small. In most the cases, images are subject to upscaling with nearest neighbor interpolation before feeding to 4-Twin Net. We assume that this technical issue also negatively affects to the performance.

Results on Grip dataset is shown in Figure 5.10. As aforementioned, Grip contains 80 base images that are not subject to rescaling, rotation or postprocessing. The only remaining artifact is edge inconsistency. However, the input size of 4-Twin Net is $64 \times 64 \times 3$, which might be insufficient to capture artifacts at edges when the forged regions are large. This limitation explains why 4-Twin Net loses its discernibility on global JPEG compression, i.e. the source and target exhibit similar visual information. For local noise adding, the accuracy levels off under 50%, which is a untypical behavior in binary classification. Since 4-Twin Net* is trained mainly on SYN which involves only *global* noise adding. Even if fine-tuned with USCISI, 4-Twin Net** is fooled with this type of attack.

On the remaining cases which leave interpolation artifacts, rotation and resizing, the two models perform pretty well. No interpolation is performed with rotation angle 0, 90, 180 and resizing factor 1.0, thus we can observe some levels off.

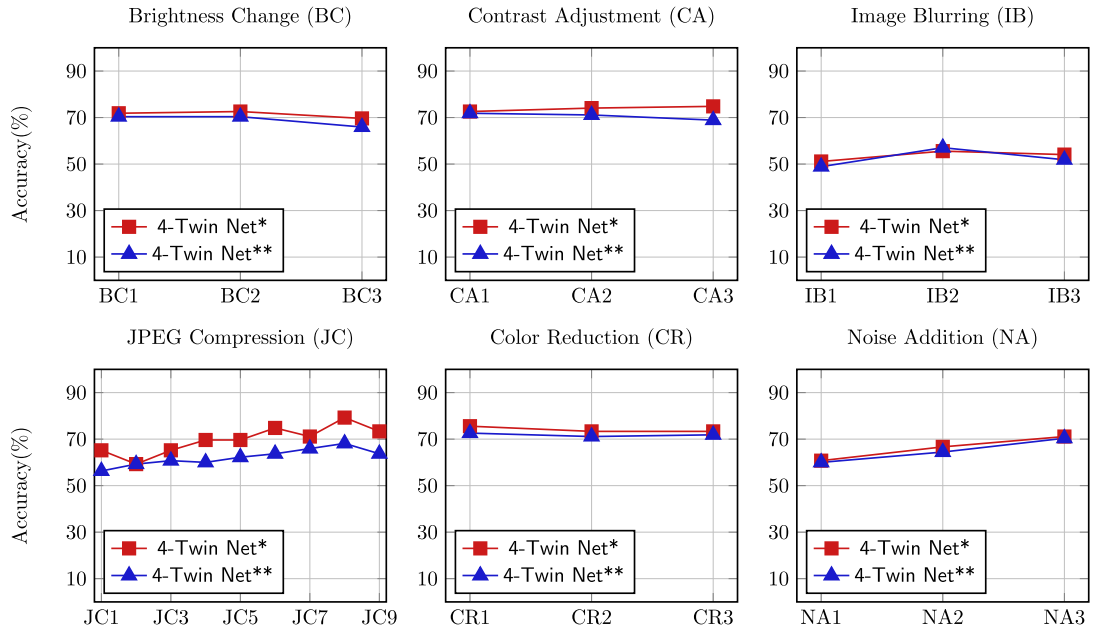


Figure 5.9: Accuracy of 4-Twin Net in CoMoFoD under different attacks. There are totally 200 images for each specific attacks. Only 135 images satisfying (1-1) condition are under test.

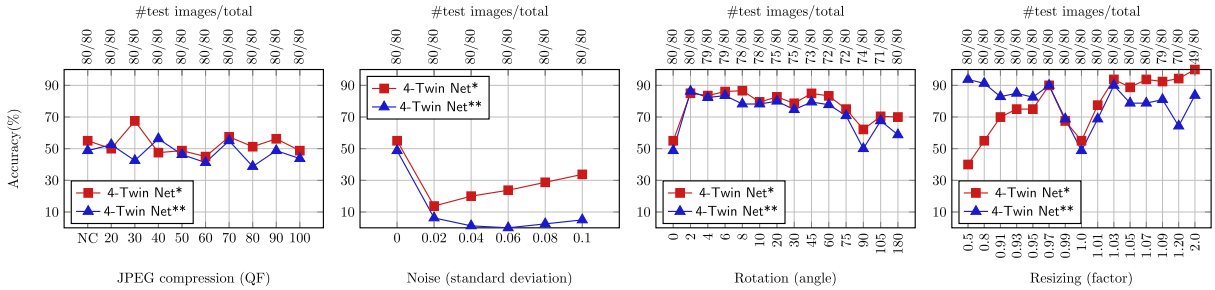


Figure 5.10: Accuracy of 4-Twin Net in Grip under different attacks. There are totally 80 images for each specific attacks. The number of images satisfying (1-1) condition is shown as top horizontal label.

5.7 Conclusions

Copy-move forgeries with careful blending make the source and target regions ambiguous. Copy-move detection algorithms have been designed for detecting and localizing duplicated regions but are not able to discern them. In this chapter, we introduce a four-stream DNN called 4-Twin Net to disambiguate source and target regions. To train 4-Twin Net, we develop an automatic data synthesizer which can generate large-scale datasets of diverged copy-move forgeries. We also propose a method to estimate the transformation matrix between source and target in case it is not given by copy-move detection methods.

Despite encouraging results, we are on the process to overcome the following issues:

- **Input shape**. Real copy-move forgeries are associated with highly non-convex shapes, while typical DNNs only accept rectangules. Currently, we employ a simple strategy: fitting a bounding box and central cropping to have shape consistency. This technical detail impacts significantly to 4-Twin Net performance.
- **Complex forms of copy-move forgeries**. We currently consider only single source and single target which are non-overlapped. Real copy-move forgeries involve more complex forms. In practice, a user might manually select two regions and use 4-Twin Net to disambiguate each pair of regions. To solve this issue automatically, we might make use of keypoint-based methods to properly pair source and target before estimating the transformation matrix.
- **End-to-end implementation**. To make 4-Twin Net an end-to-end system including copy-move detection and source-target disambiguation, we might have to implement a customized detection method which outputs two localized regions. By incorporating with this feature, we can fairly compare to BusterNet, the only available end-to-end system for such purpose.

Chapter 6

Conclusion

This doctoral study introduced several methodologies towards uncovering digital image provenance. In particular, we addressed practical concerns on digital image origin with respect to source camera and social network platforms, and the disambiguation of source and target in copy-move forgeries. Despite the fact that the proposed methods possess some improved features, they are also prone to several limitations which have been highlighted in the end of every chapter.

Methodologies proposed for image clustering by source camera have been proved to be accurate and scalable. However, performance cannot be guaranteed when the number of cameras is much larger than the average number of images per camera, and images are misaligned with respect to underlying pixel positions. One cause of the first matter is due to random splitting which separates images belonging to the same camera to different subsets, and strategic splitting might possibly mitigate this problem. On the other hand, pixel misalignment is more challenging and it demands more investigations.

Regarding to the identification of social network origin, we have considered the multiple sharing scenario where images might be shared/uploaded up to three times. Experimental observations suggest that the number of JPEG compressions has significant impact to the accuracy. Compressing an image many times results in the convergence of DCT coefficients, erasing discriminative traces. Furthermore, if the social network platform makes change on compressing and resizing policies, it is important to make our methodologies adapted to such changes.

Despite being on progress, our developed model for source-target disambiguation is a promising research direction, and in the near future, we will spend efforts to address complex forms of copy-move forgeries as well as to implement an end-to-end system of copy-move detection with source-target discernibility.

In conclusion, image provenance analysis is fundamental in digital image forensics. Its final target is to build a trustworthy profile of a digital image including information

of image origin and manipulations. Proposed methodologies in the literature are mainly associated with some assumptions and tested under limited conditions. By the advances of technologies and deep learning, the massive amount of data, it is time to explore to which extent the theoretical and practical aspects can be expanded to realistic cases.

Bibliography

[1] D. Tralic, I. Zupancic, S. Grgic, and M. Grgic. CoMoFoD – new database for copy-move forgery detection. In *Proc. of ELMAR*, pages 49–54, 2013.

[2] X. Lin and C. T. Li. Large-scale image clustering based on camera fingerprints. *IEEE Trans. on Information Forensics and Security*, 12(4):793–808, 2017.

[3] T. Gloe and R. Böhme. The 'Dresden Image Database' for benchmarking digital image forensics. In *Proc. of ACM SAC*, volume 2, pages 1585–1591, 2010.

[4] T.-T. Ng, S.-F. Chang, C.-Y. Lin, and Q. Sun. Passive-blind image forensics. In *Proc. of Multimedia Security Technologies for Digital Rights*, pages 383–412, 2006.

[5] J. R. Talburt and Y. Zhou. Chapter 11 - ISO data quality standards for master data. In *Entity Information Life Cycle for Big Data*, pages 191–205. 2015.

[6] Q.-T. Phan, G. Boato, F. G.B. De Natale. Image clustering by source camera via sparse representation. In *Proc. of MFSec*, pages 1–5, 2017.

[7] Q.-T. Phan, G. Boato, and F. G. B. De Natale. Accurate and scalable image clustering based on sparse representation of camera fingerprint. *IEEE Trans. on Information Forensics and Security*, 2019.

[8] Q.-T. Phan, C. Pasquini, G. Boato, and F. G. B. De Natale. Identifying image provenance: An analysis of mobile instant messaging apps. In *Proc. of MMSP*, pages 1–6, 2018.

[9] Q.-T. Phan, D.-T. Dang-Nguyen, G. Boato, and F. G. B. De Natale. Using LDP-TOP in video-based spoofing detection. In *Image Analysis and Processing*, pages 614–624, 2017.

[10] Q.-T. Phan, A. Budroni, C. Pasquini, F. G. B. De Natale. A hybrid approach for multimedia use verification. In *MediaEval*, 2016.

[11] Q.-T. Phan, D.-T. Dang-Nguyen, G. Boato, and F. G. B. De Natale. Face spoofing detection using LDP-TOP. In *Proc. of ICIP*, pages 404–408, 2016.

[12] R. Böhme, F. C. Freiling, T. Gloe, and M. Kirchner. Multimedia forensics is not computer forensics. In *Proc. of Computational Forensics*, pages 90–103, 2009.

[13] S. Bayram, H. Sencar, N. Memon, and I. Avcibas. Source camera identification based on CFA interpolation. In *Proc. of ICIP*, pages 69–72, 2005.

[14] A. Swaminathan, M. Wu, and K. J. R. Liu. Nonintrusive component forensics of visual sensors using output images. *IEEE Trans. on Information Forensics and Security*, 2:91–105, 2007.

[15] H. Cao and A. C. Kot. Accurate detection of demosaicing regularity for digital image forensics. *IEEE Transactions on Information Forensics and Security*, 4(4):899–910, 2009.

[16] M. Kharrazi, H. T. Sencar, and N. Memon. Blind source camera identification. In *Proc. of ICIP*, pages 709–712, 2004.

[17] O. Celiktutan, B. Sankur, and I. Avcibas. Blind identification of source cell-phone model. *IEEE Trans. on Information Forensics and Security*, 3(3):553–566, 2008.

[18] A. C. Popescu and H. Farid. Statistical tools for digital forensics. In *Information Hiding*, pages 128–147, 2005.

[19] Z. Deng, A. Gijsenij, and J. Zhang. Source camera identification using auto-white balance approximation. In *Proc. of ICCV*, pages 57–64, 2011.

[20] K. S. Choi, E. Y. Lam, and K. K. Y. Wong. Automatic source camera identification using the intrinsic lens radial distortion. *Optical Express*, 14(24):11551–11565, 2006.

[21] M. K. Johnson and H. Farid. Exposing digital forgeries through chromatic aberration. In *Proc. of MM&Sec*, pages 48–55, 2006.

[22] V. T. Lanh, S. Emmanuel, and M. S. Kankanhalli. Identifying source cell phone using chromatic aberation. In *Proc. of ICME*, 2007.

[23] A. Winkler T. Gloe, K. Borowka. Efficient estimation and large-scale evaluation of lateral chromatic aberration for digital image forensics. In *Proc. of SPIE 7541, MF&Sec II*, 2010.

[24] S. Lyu. Estimating vignetting function from a single image for image authentication. In *Proc. of MM&Sec '10*, pages 3–12, 2010.

[25] 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*, 24(3):259–263, 2017.

[26] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro. Tampering detection and localization through clustering of camera-based CNN features. In *Proc. of CVPRW*, pages 1855–1864, 2017.

[27] A. E. Dirik, H. T. Sencar, and N. Memon. Digital single lens reflex camera identification from traces of sensor dust. *IEEE Trans. on Information Forensics and Security*, 3(3):539–552, 2008.

[28] J. Lukáš, J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *IEEE Trans. on Information Forensics and Security*, 1(2):205–214, 2006.

[29] M. Chen, J. Fridrich, M. Goljan, and J. Lukáš. Determining image origin and integrity using sensor noise. *IEEE Trans. on Information Forensics and Security*, 3(1):74–90, 2008.

[30] M. Goljan and J. Fridrich. Camera identification from cropped and scaled images. In *Security, Forensics, Steganography, and Watermarking of Multimedia Contents*, 2008.

[31] M. Goljan. Digital camera identification from images – estimating false acceptance probability. In *Digital Watermarking*, pages 454–468, 2009.

[32] X. Kang, Y. Li, Z. Qu, and J. Huang. Enhancing source camera identification performance with a camera reference phase sensor pattern noise. *IEEE Trans. on Information Forensics and Security*, 7(2):393–402, 2012.

[33] C. Li. Source camera identification using enhanced sensor pattern noise. *IEEE Trans. on Information Forensics and Security*, 5(2):280–287, 2010.

[34] X. Lin and C. T. Li. Preprocessing reference sensor pattern noise via spectrum equalization. *IEEE Trans. on Information Forensics and Security*, 11(1):126–140, 2016.

[35] G. Chierchia, S. Parrilli, G. Poggi, C. Sansone, and L. Verdoliva. On the influence of denoising in prnu based forgery detection. In *Proc. of MiFor*, pages 117–122, 2010.

[36] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans. on Image Processing*, 16(8):2080–2095, 2007.

[37] G. J. Bloy. Blind camera fingerprinting and image clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(3):532–534, 2008.

[38] C.-T. Li. Unsupervised classification of digital images using enhanced sensor pattern noise. In *Proc. of ISCAS*, pages 3429–3432, 2010.

[39] C.-T. Li and X. Lin. A fast source-oriented image clustering method for digital forensics. *EURASIP Journal on Image and Video Processing*, 2017(1), 2017.

[40] R. Caldelli, I. Amerini, F. Picchioni, and M. Innocenti. Fast image clustering of unknown source images. In *Proc. of WIFS*, pages 1–5, 2010.

[41] L. Javier G. Villalba, A. L. S. Orozco, and J. R. Corripio. Smartphone image clustering. *Expert Systems with Applications*, 42(4):1927–1940, 2015.

[42] O. M. Fahmy. An efficient clustering technique for cameras identification using sensor pattern noise. In *Proc. of IWSSIP*, pages 249–252, 2015.

[43] B. B. Liu, H. K. Lee, Y. Hu, and C. H. Choi. On classification of source cameras: A graph based approach. In *Proc. of WIFS*, pages 1–5, 2010.

[44] I. Amerini, R. Caldelli, P. Crescenzi, A. Del Mastio, and A. Marino. Blind image clustering based on the normalized cuts criterion for camera identification. *Signal Processing: Image Communication*, 29(8):831–843, 2014.

[45] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva. Correlation clustering for PRNU-based blind image source identification. In *Proc. of WIFS*, pages 1–6, 2016.

[46] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva. Blind PRNU-based image clustering for source identification. *IEEE Trans. on Information Forensics and Security*, 12(9):2197–2211, 2017.

[47] C. Boididou, S. Papadopoulos, L. Apostolidis, and Y. Kompatsiaris. Learning to detect misleading content on Twitter. In *Proc. of ICMR*, pages 278–286, 2017.

[48] C. Boididou, S. E. Middleton, Z. Jin, S. Papadopoulos, D.-T. Dang-Nguyen, G. Boato, and Y. Kompatsiaris. Verifying information with multimedia content on Twitter. *Multimedia Tools and Applications*, 77(12):15545–15571, 2018.

[49] C. Boididou, S. Papadopoulos, D.-T. Dang-Nguyen, G. Boato, M. Riegler, A. Petlund, and Y. Kompatsiaris. Verifying multimedia use at MediaEval 2016. In *MediaEval*, 2016.

[50] Q.-T. Phan, A. Budroni, C. Pasquini, and F.G.B. De Natale. A hybrid approach for multimedia use verification. In *MediaEval*, 2016.

[51] M. Zampoglou, S. Papadopoulos, and K. Yiannis. Large-scale evaluation of splicing localization algorithms for web images. *Multimedia Tools and Applications*, 76(4):4801–4834, 2017.

[52] C. Maigrot, V. Claveau, E. Kijak, and R. Sicre. MediaEval 2016: A multimodal system for the verifying multimedia use task. In *MediaEval*, 2016.

[53] G. K. Wallace. The JPEG still picture compression standard. *IEEE Trans. on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[54] R. Caldelli, R. Becarelli, and I. Amerini. Image origin classification based on social network provenance. *IEEE Trans. on Information Forensics and Security*, 12(6):1299–1308, 2017.

[55] O. Giudice, A. Paratore, M. Moltisanti, and S. Battiato. A classification engine for image ballistics of social data. In *Proc. of ICIAP*, pages 625–636, 2017.

[56] I. Amerini, T. Uricchio, and R. Caldelli. Tracing images back to their social network of origin: A CNN-based approach. In *Proc. of WIFS*, pages 1–6, 2017.

[57] M. C. Stamm, M. Wu, and K. J. R. Liu. Information forensics: An overview of the first decade. *IEEE Access*, 1:167–200, 2013.

[58] B. Bayar and M. C. Stamm. A generic approach towards image manipulation parameter estimation using convolutional neural networks. In *Proc. of IH&MMSec*, pages 147–157, 2017.

[59] M. Boroumand and J. Fridrich. Scalable processing history detector for JPEG images. *Electronic Imaging*, 2017(7):128–137, 2017.

[60] C.-H. Choi, H.-Y. Lee, and H.-K. Lee. Estimation of color modification in digital images by CFA pattern change. *Forensic science international*, 226:94–105, 01 2013.

[61] J. Hou and H. Lee. Detection of hue modification using photo response nonuniformity. *IEEE Trans. on Circuits and Systems for Video Technology*, 27(8):1826–1832, 2017.

[62] H. Farid. Blind inverse gamma correction. *IEEE Trans. on Image Processing*, 10(10):1428–1433, 2001.

[63] M. C. Stamm and K. J. R. Liu. Forensic estimation and reconstruction of a contrast enhancement mapping. In *Proc. of ICASSP*, pages 1698–1701, 2010.

[64] X. Zhang and S. Lyu. Blind estimation of pixel brightness transform. In *Proc. of ICIP*, pages 4472–4476, 2014.

[65] L. Wen, H. Qi, and S. Lyu. Contrast enhancement estimation for digital image forensics. *ACM Trans. Multimedia Computing, Communications, and Applications*, 14(2):49:1–49:21, 2018.

[66] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva. Reverse engineering of double compressed images in the presence of contrast enhancement. In *Proc. of MMSP*, pages 141–146, 2013.

[67] A. C. Gallagher. Detection of linear and cubic interpolation in JPEG compressed images. In *Proc. of CRV*, pages 65–72, 2005.

[68] A. C. Popescu and H. Farid. Exposing digital forgeries by detecting traces of resampling. *IEEE Trans. on Signal Processing*, 53(2):758–767, 2005.

[69] M. Kirchner. Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue. In *Proc. of MM&Sec*, pages 11–20, 2008.

[70] T. Bianchi and A. Piva. Reverse engineering of double JPEG compression in the presence of image resizing. In *Proc. of WIFS*, pages 127–132, 2012.

[71] B. Bayar and M. C. Stamm. On the robustness of constrained convolutional neural networks to JPEG post-compression for image resampling detection. In *Proc. of ICASSP*, pages 2152–2156, 2017.

[72] X. Kang, M. C. Stamm, A. Peng, and K. J. R. Liu. Robust median filtering forensics using an autoregressive model. *IEEE Trans. on Information Forensics and Security*, 8(9):1456–1468, 2013.

[73] D. T. Dang-Nguyen, I. D. Gebru, V. Conotter, G. Boato, and F. G. B. De Natale. Counter-forensics of median filtering. In *Proc. of MMSP*, pages 260–265, 2013.

[74] D. Kim, H. Jang, S. Mun, S. Choi, and H. Lee. Median filtered image restoration and anti-forensics using adversarial networks. *IEEE Signal Processing Letters*, 25(2):278–282, 2018.

[75] Z. Fan and R. L. de Queiroz. Identification of bitmap compression history: JPEG detection and quantizer estimation. *IEEE Trans. on Image Processing*, 12(2):230–235, 2003.

[76] D. Fu, Y.Q. Shi, and W. Su. A generalized Benford's law for JPEG coefficients and its applications in image forensics. *Proc. of SPIE 6505, Security, Steganography, and Watermarking of Multimedia Contents IX*, 6505, 02 2007.

[77] J. Lukáš and J. Fridrich. Estimation of primary quantization matrix in double compressed JPEG images. In *Proc. of DFRWS*, 2003.

[78] T. Pevný and J. Fridrich. Estimation of primary quantization matrix for steganalysis of double-compressed JPEG images. In *Proc. of SPIE*, volume 6819, pages 6819 – 6819 – 13, 2008.

[79] T. Bianchi and A. Piva. Detection of nonaligned double JPEG compression based on integer periodicity maps. *IEEE Trans. on Information Forensics and Security*, 7(2):842–848, 2012.

[80] Y. Wu, W. Abd-Almageed, and P. Natarajan. BusterNet: Detecting copy-move image forgery with source/target localization. In *Proc. ECCV*, pages 170–186, 2018.

[81] M. Chen, J. Fridrich, and M. Goljan. Digital imaging sensor identification (further study). In *Proc. of SPIE Electronic Imaging, Security, Steganography, Watermarking of Multimedia Contents IX*, volume 6505, 2007.

[82] M. Goljan, J. Fridrich, and T. Filler. Managing a large database of camera fingerprints. In *Proc. of SPIE 7541, Media Forensics and Security II*, volume 7541, pages 7541–7541–12, 2010.

[83] T. Gloe, S. Pfennig, and M. Kirchner. Unexpected artefacts in prnu-based camera identification: A 'dresden image database' case-study. In *Procs. of Multimedia and Security*, pages 109–114, 2012.

[84] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013.

[85] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès. Robust subspace clustering. *The Annals of Statistics*, 42, 2014.

[86] Y.-X. Wang and H. Xu. Noisy sparse subspace clustering. *Journal of Machine Learning Research*, 17(12):1–41, 2016.

[87] D. L. Donoho. For most large underdetermined systems of equations, the minimal ℓ_1-norm near-solution approximates the sparsest near-solution. In *Technical report (Stanford University. Dept. of Statistics)*. 2004.

[88] A. Y. Yang, Z. Zhou, A. G. Balasubramanian, S. S. Sastry, and Y. Ma. Fast ℓ_1-minimization algorithms for robust face recognition. *IEEE Trans. on Image Processing*, 22(8):3234–3246, 2013.

[89] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Machine Learning*, 3(1):1–122, 2011.

[90] D. Shullani, M. Fontani, M. Iuliani, O. A. Shaya, and A. Piva. VISION: a video and image dataset for source identification. *EURASIP Journal on Information Security*, 2017(1), 2017.

[91] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. of the 14th NIPS*, pages 849–856, 2001.

[92] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[93] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Computational and Applied Mathematics*, 20:53–65, 1987.

[94] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. *The Royal Statistical Society Series B (Statistical Methodology)*, 63:411–423, 2000.

[95] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[96] S. Rosset and J. Zhu. Piecewise linear regularized solution paths. *The Annals of Statistics*, 35(3):1012–1030, 2007.

[97] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the 2nd KDD*, pages 226–231, 1996.

[98] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. of ACM SIGMOD MOD*, pages 103–114, 1996.

[99] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proc. of ACM SIGMOD MOD*, pages 73–84, 1998.

[100] X. Peng, L. Zhang, and Z. Yi. Scalable sparse subspace clustering. In *Proc. of CVPR*, pages 430–437, 2013.

[101] X. Peng, H. Tang, L. Zhang, Z. Yi, and S. Xiao. A unified framework for representation-based subspace clustering of out-of-sample and large-scale data. *IEEE Trans. on Neural Networks and Learning Systems*, 27(12):2499–2512, 2016.

[102] C. You, D. P. Robinson, and R. Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proc. of CVPR*, pages 3918–3927, 2016.

[103] C. You, D. P. Robinson, and R. Vidal. Provable self-representation based outlier detection in a union of subspaces. In *Proc. of CVPR*, pages 4323–4332, 2017.

[104] J. Fridrich and M. Goljan. Derivation of ROCs for composite fingerprints and sequential trimming. In *Technical Report (Binghamton University)*. 2010.

[105] J. Lukáš, J. Fridrich, and M. Goljan. Determining digital image origin using sensor imperfections. In *Proc. of SPIE - The International Society for Optical Engineering*, volume 5685, pages 5685–5685–12, 2005.

[106] J. Eklann. Source camera classification and clustering from sensor pattern noise. Master's thesis, Dept. Appl. Inf. Tech., Chalmers Uni. Tech., Gothenburg, Sweden, 2012.

[107] J. Besag. On the statistical analysis of dirty pictures. *The Royal Statistical Society B*, 48(3):48–259, 1986.

[108] D. Huang, J.-H. Lai, C.-D. Wang. Combining multiple clusterings via crowd agreement estimation and multi-granularity link analysis. *Neurocomputing*, 170:240–250, 2015.

[109] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[110] LAPACK. `http://www.netlib.org/lapack/`. Accessed: 2017-08-31.

[111] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. RAISE: A raw images dataset for digital image forensics. In *Proc. of ACM MMSys*, pages 219–224, 2015.

[112] R. Li, C. T. Li, and Y. Guan. A compact representation of sensor fingerprint for camera identification and fingerprint matching. In *Proc. of ICASSP*, pages 1777–1781, 2015.

[113] Q. Rao and J. Wang. Suppressing random artifacts in reference sensor pattern noise via decorrelation. *IEEE Signal Processing Letters*, 24(6):809–813, 2017.

[114] R. Li, C.-T. Li, and Y. Guan. Inference of a compact representation of sensor fingerprint for source camera identification. *Pattern Recognition*, 74:556–567, 2018.

[115] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli. Compressed fingerprint matching and camera identification via random projections. *IEEE Trans. on Information Forensics and Security*, 10(7):1472–1485, 2015.

[116] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli. Large-scale image retrieval based on compressed camera identification. *IEEE Trans. on Multimedia*, 17(9):1439–1449, 2015.

[117] D. Cozzolino and L. Verdoliva. Noiseprint: a CNN-based camera model fingerprint. *CoRR*, abs/1808.08396, 2018.

[118] H. Farid. Photo forensics. *MIT Press*, 2016.

[119] B. Mahdian and S. Saic. Detecting double compressed JPEG images. In *Proc. of ICDP*, pages 1–6, 2009.

[120] T. Pevny and J. Fridrich. Detection of double-compression in JPEG images for applications in steganography. *IEEE Trans. on Information Forensics and Security*, 3(2):247–258, 2008.

[121] F. Huang, J. Huang, and Y. Q. Shi. Detecting double JPEG compression with the same quantization matrix. *IEEE Trans. on Information Forensics and Security*, 5(4):848–856, 2010.

[122] P. Alvarez. Using extended file information (EXIF) file headers in digital evidence analysis. *Int. Journal of Digital Evidence*, 2, 2004.

[123] E. Kee, M. K. Johnson, and H. Farid. Digital image authentication from JPEG headers. *IEEE Trans. on Information Forensics and Security*, 6(3):1066–1075, 2011.

[124] B. Bayar and M. C. Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proc. of IH&MMSec*, pages 5–10, 2016.

[125] Y. Rao and J. Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *Proc. of WIFS*, pages 1–6, 2016.

[126] D. Cozzolino, G. Poggi, and L. Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: An application to image forgery detection. In *Proc. of IH&MMSec*, pages 159–164, 2017.

[127] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Two-stream neural networks for tampered face detection. In *Proc. of CVPRW*, pages 1831–1839, 2017.

[128] O. Mayer and M. C. Stamm. Learned forensic source similarity for unknown camera models. In *Proc. of ICASSP*, pages 2012–2016, 2018.

[129] 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 Comunication and Image Representation*, 49:153–163, 2017.

[130] V. Sedighi and J. Fridrich. Histogram layer, moving convolutional neural networks towards feature-based steganalysis. *Electronic Imaging*, 2017(7):50–55, 2017.

[131] C. F. Tsang, J. Fridrich. Steganalyzing images of arbitrary size with CNNs. In *Proc. of Media Watermarking, Security, and Forensics*, 2018.

[132] G. Schaefer and M. Stich. Ucid - an uncompressed colour image database. In *Proc. of SRMAM*, volume 5307, pages 472–480, 2004.

[133] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou. An evaluation of popular copy-move forgery detection approaches. *IEEE Trans on Information Forensics and Security*, 7(6):1841–1854, 2012.

[134] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a ''siamese'' time delay neural network. In *Proc. of NIPS*, pages 737–744, 1993.

[135] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, pages 770–778, 2016.

[136] M. Huh, A. Liu, A. Owens, and A. A. Efros. Fighting fake news: Image splice detection via learned self-consistency. In *Proc. of The ECCV*, 2018.

[137] G. Koch, C. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *Proc. of ICML DL workshop*, 2015.

[138] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proc. of CVPR*, pages 539–546, 2005.

[139] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of ICML*, pages 448–456, 2015.

[140] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, pages 417–441, 1933.

[141] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Proc. of CVPR*, pages 3485–3492, 2010.

[142] T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick. Microsoft COCO: Common objects in context. In *Proc. of ECCV*, pages 740–755, 2014.

[143] J. Dong, W. Wang, and T. Tan. CASIA image tampering detection evaluation database. In *Proc. of CS and Int. Conf. on SIP*, pages 422–426, July 2013.

[144] D. Cozzolino, G. Poggi, and L. Verdoliva. Copy-move forgery detection based on patchmatch. In *Proc. of ICIP*, pages 5312–5316, 2014.

[145] W. Cheney and A. A. Goldstein. Proximity maps for convex sets. *The American Mathematical Society*, 10(3):448–450, 1959.

[146] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

Chapter 7

Supplemental Materials

7.1 PRNU estimation

Given a grayscale image $\mathbf{Y}$, its noise residual composes PRNU component modulated by image content $\mathbf{Y}$, and white Gaussian noise $\mathbf{\Xi}$:

$$\mathbf{W} = \mathbf{YK} + \mathbf{\Xi}. \tag{7.1}$$

The white Gaussian noise at each pixel j is assumed to have variance σ^2 and be independent with $\mathbf{YK}$, i.e. $\mathbf{\Xi}[j] \sim \mathcal{N}(0, \sigma^2)$. Without loss of generality, the index j is omitted for simplicity, and the following derivations are interpreted as pixel-wise operations. For each of N images taken by the same camera, Eq. (7.1) becomes:

$$\frac{\mathbf{W}_i}{\mathbf{Y}_i} - \mathbf{K} = \frac{\mathbf{\Xi}_i}{\mathbf{Y}_i}, \quad 1 \leq i \leq N.$$

The variance of white Gaussian noise from every image of the same camera is assumed to be stationary, $\frac{\mathbf{\Xi}_i}{\mathbf{Y}_i} \sim N(0, \frac{\sigma^2}{\mathbf{Y}_i^2})$. Likelihood of $\frac{\mathbf{W}_i}{\mathbf{Y}_i} - \mathbf{K}$ is:

$$\Pr\left[\frac{\mathbf{W}_i}{\mathbf{Y}_i} - \mathbf{K}\right] = \frac{\mathbf{Y}_i}{\sqrt{2\pi}\sigma} e^{\frac{-\left(\frac{\mathbf{W}_i}{\mathbf{Y}_i} - \mathbf{K}\right)^2}{2\sigma^2/\mathbf{Y}_i^2}}.$$

Likelihood of observing $\frac{\mathbf{W}_1}{\mathbf{Y}_1}, \frac{\mathbf{W}_2}{\mathbf{Y}_2}, \ldots, \frac{\mathbf{W}_N}{\mathbf{Y}_N}$ giving $\mathbf{K}$ is:

$$\Pr\left[\frac{\mathbf{W}_1}{\mathbf{Y}_1}, \frac{\mathbf{W}_2}{\mathbf{Y}_2}, \ldots, \frac{\mathbf{W}_N}{\mathbf{Y}_N} \,\middle|\, \mathbf{K}\right] = \prod_{i=1}^{N} \frac{\mathbf{Y}_i}{\sqrt{2\pi}\sigma} e^{\frac{-\left(\frac{\mathbf{W}_i}{\mathbf{Y}_i} - \mathbf{K}\right)^2}{2\sigma^2/\mathbf{Y}_i^2}}. \tag{7.2}$$

Maximizing likelihood is cast to maximizing logarithm of likelihood (LLH). By taking logarithm of Eq. (7.2):

$$\mathrm{LLH}\left[\frac{\mathbf{W}_1}{\mathbf{Y}_1}, \frac{\mathbf{W}_2}{\mathbf{Y}_2}, \ldots, \frac{\mathbf{W}_N}{\mathbf{Y}_N} \,\middle|\, \mathbf{K}\right] = \sum_{i=1}^{N} \log\left(\frac{\mathbf{Y}_i}{\sqrt{2\pi}\sigma}\right) - \sum_{i=1}^{N} \frac{-\left(\frac{\mathbf{W}_i}{\mathbf{Y}_i} - \mathbf{K}\right)^2}{2\sigma^2/\mathbf{Y}_i^2}. \tag{7.3}$$

By taking the partial derivative of Eq. (7.3) and setting it as 0

$$\frac{\partial \text{LLH}\left[\frac{\mathbf{W}_1}{\mathbf{Y}_1}, \frac{\mathbf{W}_2}{\mathbf{Y}_2}, \ldots, \frac{\mathbf{W}_N}{\mathbf{Y}_N} \,\middle|\, \mathbf{K}\right]}{\partial \mathbf{K}} = \sum_{i=1}^{N} \frac{\left(\frac{\mathbf{W}_i}{\mathbf{Y}_i} - \mathbf{K}\right)}{\sigma^2/\mathbf{Y}_i^2} = 0,$$

we can then solve for $\mathbf{K}$:

$$\mathbf{K} = \frac{\sum_{i=1}^{N} \mathbf{W}_i \mathbf{Y}_i}{\sum_{i=1}^{N} \mathbf{Y}_i^2}.$$

7.2 Derivation of V Update in Algorithm 1

At each iteration of CONSTRAINED_LASSO, $\mathbf{V}$ is updated by:

$$\mathbf{V} \;=\; \arg\min_{\mathbf{V}} f(\mathbf{V}),$$

where

$$\begin{aligned}
f(\mathbf{V}) &= \gamma\|\mathbf{V}\|_1 + \langle \mathbf{\Lambda}, \mathbf{Z} - \mathbf{V}\rangle + \frac{\eta}{2}\|\mathbf{Z} - \mathbf{V}\|_F^2. \\
\frac{\partial f}{\partial \mathbf{V}_{ij}} &= \gamma \frac{\partial |\mathbf{V}_{ij}|}{\partial \mathbf{V}_{ij}} + \eta \mathbf{V}_{ij} - \eta\left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right) \\
&= \begin{cases} -\gamma + \eta \mathbf{V}_{ij} - \eta\left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right), & \mathbf{V}_{ij} < 0 \\ \gamma + \eta \mathbf{V}_{ij} - \eta\left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right), & \mathbf{V}_{ij} > 0 \\ \text{undefined}, & \mathbf{V}_{ij} = 0 \end{cases}. \\
\frac{\partial f}{\partial \mathbf{V}_{ij}} &= 0 \text{ then} \\
\mathbf{V}_{ij} &= S_{\frac{\gamma}{\eta}}\left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right) \\
&= \begin{cases} \frac{\gamma}{\eta} + \mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}, & \text{if} \quad \mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta} < -\frac{\gamma}{\eta} \\ \frac{-\gamma}{\eta} + \mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}, & \text{if} \quad \mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta} > \frac{\gamma}{\eta} \\ 0, & \text{if} \quad \left|\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right| \le \frac{\gamma}{\eta} \end{cases}.
\end{aligned}$$

Solution $\mathbf{V} \in \mathbb{R}^{n\times n}$ might violate two constraints in Equation (3.5). Denote $\mathcal{C}_1$ the set of all zero-diagonal and $\mathcal{C}_2$ the set of non-negative matrices. $\mathcal{C}_1, \mathcal{C}_2$ are convex. To impose the two constraints on $\mathbf{V}$, it is equivalent to find $\mathbf{V}^{(1,2)} \in \mathcal{C}_1 \cap \mathcal{C}_2$ that minimizes f. This can be obtained via von Neumann's alternating projections [145]: first Euclidean projection onto $\mathcal{C}_1$, and second Euclidean projection onto $\mathcal{C}_2$. Since $\mathbf{V} \in \mathbb{R}^{n\times n}$ is a minimizer of f, $\mathbf{V}^{(1,2)}$ can be obtained by two successive projections:

$$\mathbf{V}^{(1)} = \arg\min_{\tilde{\mathbf{V}} \in \mathcal{C}_1} \|\mathbf{V} - \tilde{\mathbf{V}}\|_F^2,$$

$$\mathbf{V}^{(1,2)} = \arg\min_{\tilde{\mathbf{V}} \in \mathcal{C}_2} \|\mathbf{V}^{(1)} - \tilde{\mathbf{V}}\|_F^2.$$

The two projections are implemented element-wise as in Equation (3.7) and Equation (3.8), respectively.

7.3 $\mathcal{F}$-measure and ARI In The Presence of Outliers

In the presence of outliers (unclustered images), if they are considered as a normal cluster (as the dataset has no outlier), we can compute $\mathcal{F}$-measure and ARI canonically. Otherwise, we can treat outliers differently in the computation of True Positive ($\overline{TP}$) and False Positive ($\overline{FP}$):

- $\overline{TP}$: the number of image pairs from the same cluster which are assigned to the same cluster, *excluding outliers*.
- $\overline{FP}$: the number of image pairs from different clusters which are assigned to the same cluster, *excluding outliers*.

The main reason for this modification is that outliers are not considered as a normal cluster. On the other hand, TN and FN should be computed canonically.

Denote $U = \{U_1, U_2, \ldots, U_{L_g}\}$ the ground-truth clusters, and $V = \{V_0, V_1, \ldots, V_{L_p}\}$ the predicted clusters. V_0 is a special cluster containing unclustered fingerprints (highlighted in red), if any. The contingency matrix of U, V has the form:

	U_1	U_2	$\ldots$	U_{L_g}	$c_{i\cdot} = \sum_{j=1}^{L_g} c_{ij}$
V_0	c_{01}	c_{02}	$\ldots$	c_{0L_g}	$c_{0\cdot}$
V_1	c_{11}	c_{12}	$\ldots$	c_{1L_g}	$c_{1\cdot}$
$\vdots$	$\vdots$	$\vdots$	$\ldots$	$\vdots$	$\vdots$
V_{L_p}	c_{L_p1}	c_{L_p2}	$\ldots$	$c_{L_pL_g}$	$c_{L_p\cdot}$
$c_{\cdot j} = \sum_{i=0}^{L_p} c_{ij}$	$c_{\cdot 1}$	$c_{\cdot 2}$	$\ldots$	$c_{\cdot L_g}$	

where c_{ij} is the number common images between V_i and U_j.

We can compute all basic measures as:

$$TP = \sum_{i=0}^{L_p}\sum_{j=1}^{L_g}\binom{c_{ij}}{2}, \quad FP = \sum_{i=0}^{L_p}\binom{c_{i\cdot}}{2} - TP,$$

$$\overline{TP} = \sum_{i=1}^{L_p}\sum_{j=1}^{L_g}\binom{c_{ij}}{2}, \quad \overline{FP} = \sum_{i=1}^{L_p}\binom{c_{i\cdot}}{2} - \overline{TP},$$

$$FN = \sum_{j=1}^{L_g}\binom{c_{\cdot j}}{2} - TP, \quad TN = \binom{n}{2} - TP - FP - FN$$

$\mathcal{F}$-measure is computed based on precision ($\mathcal{P}$) and recall ($\mathcal{R}$), taking $\overline{TP}$ and $\overline{FP}$ into account:

$$\mathcal{P} = \frac{\overline{TP}}{\overline{TP} + \overline{FP}}, \quad \mathcal{R} = \frac{\overline{TP}}{\overline{TP} + FN}, \quad \mathcal{F} = 2 \cdot \frac{\mathcal{P} \cdot \mathcal{C}}{\mathcal{P} + \mathcal{C}}$$

Rand Index (RI) and ARI are computed as:

$$\text{RI} = \frac{\overline{TP} + TN}{\overline{TP} + TN + \overline{FP} + FN}, \quad \text{ARI} = \frac{\text{RI} - \mathbf{E}[\text{RI}]}{1 - \mathbf{E}[\text{RI}]},$$

where $\mathbf{E}[\text{RI}]$ is the expected value of RI and is computed based on the expected value of $\overline{TP}$ and TN.

$$\mathbf{E}[\text{RI}] = \frac{\mathbf{E}[\overline{TP}] + \mathbf{E}[TN]}{\overline{TP} + TN + \overline{FP} + FN}$$

$$\begin{aligned}
\mathbf{E}[\overline{TP}] &= \sum_{i=1}^{L_p} \binom{c_{i\cdot}}{2} \sum_{j=1}^{L_g} \binom{c_{\cdot j}}{2} \Big/ \binom{n - c_{0\cdot}}{2} \\
\mathbf{E}[TN] &= \binom{n}{2} - \sum_{i=0}^{L_p} \binom{c_{i\cdot}}{2} - \sum_{j=1}^{L_g} \binom{c_{\cdot j}}{2} \\
&+ \sum_{i=0}^{L_p} \binom{c_{i\cdot}}{2} \sum_{j=1}^{L_g} \binom{c_{\cdot j}}{2} \Big/ \binom{n}{2}.
\end{aligned}$$

The readers can refer to [146] for more details of ARI computation. ca on ma