DOTTORATO DI RICERCA IN INFORMATICA
IX CICLO
UNIVERSITA' DEGLI STUDI DI SALERNO

Forensic Analysis for Digital Images

Maurizio Cembalo

November, 2010

PhD Program Chair
Prof.ssa
Margherita Napoli

Supervisor
Prof.
Alfredo De Santis

1. PhD Program Chair: Prof.ssa Margherita Napoli

2. Ph.D Committee: Prof. Alfredo De Santis, Prof. Marco Faella, Prof. Domenico Talia

3. Supervisor: Prof. Alfredo De Santis

Day of the defense: April 29th, 2011

Contents

List of Figures

List of Tables

Introduction

Nowadays, taking and sharing digital pictures is becoming a very popular activity. This is witnessed by the explosive growth of the digital cameras market: e.g., more than one billion of digital cameras have been produced and shipped in 2010 (5). A consequence of this trend is that also the number of crimes involving digital pictures increases, either because pictures are part of the crime (e.g., exchanging pedopornographic pictures) or because their analysis may reveal some important clue about the author of the crime.

The highly technical nature of computer crimes facilitated a wholly new branch of forensic science called digital forensics. The Digital Forensic Science involves processes such as acquisition of data from an electronic source, analysis of the acquired data, extraction of evidence from the data, and the preservation and presentation of the evidence. Digital Imaging Forensics is a specialization of the Digital Forensics which deals with digital images. One of the many issues that the Digital Imaging Forensics tries to deal with is the *source camera identification problem*, i.e., establish if a given image has been taken by a given digital camera. Today this is a practical and important problem aiming to identify reliably the imaging device that acquired a particular digital image. Techniques to authenticate an electronic image are especially important in court. For example, identifying the source device could establish the origin of images presented as evidence. In a prosecution for child pornography, for example, it could be desirable that one could prove that certain imagery was obtained with a specific camera and is thus not an image generated by a computer, given that "virutal images"

are not considered offense. As electronic images and digital video replace their analog counterparts, the importance of reliable, inexpensive, and fast identification of the origin of a particular image will increase.

The identification of a source camera of an image is a complex issue which requires the understanding of the several steps involved in the creation of the digital photographic representation of a real scene. In particular, it is necessary to understand how the digital images are created, which are the processes which create (and therefore affect) the creation of the digital data, starting from the real scene. Moreover, it is necessary to point out the factors which can be used to support the camera identification and, may be even more important, which are the factors which can tamper the photos and prevent (maliciously or not) the camera identification.

Many identification techniques have been proposed so far in literature. All these techniques generally work by using the sensor noise (an unexpected variation of the digital signal) left by a digital sensor when taking a picture as a fingerprint for identifying the sensor. These studies are generally accompanied with tests proving the effectiveness of these techniques, both in terms of False Acceptance Rate (FAR) and False Rejection Rate (FRR).

Unfortunately, most of these contributions do not take into consideration that, in practice, the images that are shared and exchanged over the Internet have often been pre-processed. Instead, it is a common practice to assume that the images to be examined are unmodified or, at most, to ignore the effects of the pre-processing.

Even without considering the case of malicious users that could intentionally process a picture in order to fool the existing identification techniques, this assumption is unrealistic for at least two reasons. The first is that, as previously mentioned, almost all current photo-managing software offers several functions for adjusting, sometimes in a "magic" way (see the "I'm feeling lucky" function on Google Picasa (19)) different characteristics of a picture. The second reason can be found in the way the images are

managed by some of the most important online social network (OSN) and online photo sharing (OPS) sites. These services usually make several modifications to the original photos before publishing them in order to either improve their appearance or reduce their size.

In this thesis we have first implemented the most prominent source camera identification technique, proposed by Lukáš *et al.* and based on the Photo-Response Non-Uniformity. Then, we present a new identification technique that use a SVM (Support Vector Macchine) classifier to associate photos to the right camera. Both our implementation of Lukáš *et al.* technique and our SVM technique have been extensively tested on a test-sample of nearly 2500 images taken from 8 different cameras. The main purpose of the experiments conducted is to see how these techniques performs in presence of pre-processed images, either explicit modified by a user with photo management tools or modified by OSNs and OPSs services without user awareness.

The results confirm that, in several cases, the method by Lukáš *et al.* and our SVM technique is resilient to the modifications introduced by the considered image-processing functions. However, in the experiments it has been possible to identify several cases where the quality of the identification process was deteriorated because of the noise introduced by the image-processing. In addition, when dealing with Online Social Networks and Online Photo Sharing services, it has been noted that some of them process and modify the uploaded pictures. These modifications make ineffective, in many cases, the method by Lukáš *et al.* while SVM technique performs slightly better.

In Chapter 1 some basic definitions about the forensic analysis and digital evidences, with particular attention to digital imaging forensic, are provided. Chapter 2 describes the process of creating an image and the main types of noise. In Chapter 3.1 definitions about source camera identification problem are provided as well as a brief review of the current literature on this topic. Moreover, details about the identification

technique presented by Lukáš *et al.* and the SVM classifier are introduced. In Chapter 4 are explained the conditions under which the tests have been conducted, the kinds and the methodology of experiments. Moreover, the results of several tests of the two techniques are presented. In the experiments, the performance of the techniques, when applied to the identification of the cameras used to take both modified and unmodified pictures, is compared. Finally, Chapter 5 presents some concluding remarks. In Appendix A are presented the confusion matrices for all experiments conducted with the SVM classifier. The MATLab source code for our software implementation are published in Appendix B.

Chapter 1

Forensic analysis and digital evidences

Digital forensics is a branch of forensic science concerned with the use of digital information (produced, stored and transmitted by computers) as source of evidence in investigations and legal proceedings. In the Digital Forensic Research Workshop, digital forensics has been defined as

> "The use of scientifically derived and proven methods toward the preservation, validation, identication, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations." (16)

Investigative process of digital forensics can be divided into four major stages:

- Preservation. The Preservation stage corresponds to freezing the crime scene. It consists in stopping or preventing any activities that can damage digital information being collected. Preservation involves operations such as preventing people

from using computers during collection, stopping ongoing deletion processes, and choosing the safest way to collect information.

- Collection. The Collection stage consists in finding and collecting digital information that may be relevant to the investigation. Since digital information is stored in computers, collection of digital information means either collection of the equipment containing the information, or recording the information on some medium. Collection may involve removal of personal computers from the crime scene, copying or printing out contents of files from a server, recording of network traffic, and so on.

- Examination. The Examination stage consists in an in-depth systematic search of evidence relating to the incident being investigated. The output of the examination phase are data objects found in the collected information. They may include log files, data files containing specific phrases, timestamps, and so on.

- Analysis. The aim of analysis is to draw conclusions based on evidence found.

The aims of preservation and collection are twofold. First, they aim to provide the examination and analysis stages with as much relevant information as possible. Second, they aim to ensure integrity of the collected information.

1.1 Digital evidences

Digital evidence is any probative information stored or transmitted in digital form that a party to a court case may use at trial. Digital forensic evidence consists of exhibits, each one consisting of a sequence of bits, presented by witnesses in a legal matter, to help jurors to establish the facts of the case and then supporting or refuting legal theories of the case. The exhibits should be introduced and presented and/or challenged by properly qualified people using a properly applied methodology that addresses the

legal theories at issue. Before accepting digital evidence a court will determine if the evidence is relevant, whether it is authentic, if it is hearsay and whether a copy is acceptable or the original is required (7).

The use of digital evidence has increased in the past few decades as courts have allowed the use of e-mails, images, files stored on computer hard drive, digital video, digital audio, network packets transmitted over local area network (33).

When a scientific evidence is given, it is possible that a qualified expert may have based his findings on a novel scientific theory that lacks sufficient experimental support to draw reliable conclusions. Then, it may be necessary to specify a number of non-mandatory, non-exclusive criteria for determining scientific validity such as the following ones:

- whether the theory or technique employed by the expert can be (and has been) tested;
- the known or potential rate of error associated with the theory or technique;
- the existence and maintenance of standards controlling the technique's operation;
- whether the theory or technique have been subjected to peer review and publication;
- whether the theory or technique enjoys widespread acceptance because, as said by the Judge in (15) ,"a *known* technique that has been able to attract only minimal support within the community, may properly be viewed with skepticism".

Depending on what facts the digital evidence is supposed to prove, it can fall into different classes of evidence:

- digital images or software presented in court to prove the fact of possession are real evidence;
- e-mail messages presented as proof of their content are documentary evidence;

- log files, file time stamps, all sorts of system information used to reconstruct sequence of events are circumstantial evidence;

The use of digital information in legal disputes is complicated by a number of technical problems, which reduce weight of computer based evidence or even make it irrelevant. The following ones are some common problems:

Anonymity of digital information. Digital information generated, stored, and transmitted between computing devices does not bear any physical imprints connecting it to the individual who caused its generation.

Context of digital information. Digital information is a sequence of digits encoding some knowledge. The encoding, and hence the meaning of digits is determined by the context in which the information is produced and used. Before inferences can be made, the context determining the meaning of information must be clarified.

Automated interpretation of digital information. Manual interpretation of the digital information can be extremely labor consuming or even impossible. A precondition for the use of any automated tool for interpreting digital information is the assurance that the tool gives correct interpretation of the information.

Danger of damaged information. Digital information stored on magnetic and optical media can be damaged by a variety of causes. Unlike other types of evidential material, digital information is highly sensitive to minor changes. A single bit change may cause dramatic change in its interpretation. At the same time, minor changes may be very hard to detect in a large quantity of digital information, particularly if the damaged information has valid interpretation.

1.2 Digital Imaging Forensic

In the analog world, photographic images have generally been accepted as *proof of occurrence* of the depicted events. But today, the creation and manipulation of images are made simple by digital processing tools that are widely available and easily to use. As a result, we can no longer take the authenticity of any images, analog or digital, for granted. This is especially true when they are used as digital evidence or *photographic evidence* by law enforcement agencies. In this context, the field of image forensics is concerned with uncovering underlying facts about an image: by using its techniques, we attempt to provide authoritative answers to several questions about the content and source of an image. Is this an "original", or was it created by cut and paste operations from different images? Was it captured by a camera manufactured by vendor X or vendor Y? Did it originate from camera X as claimed? At time Y? At location Z? Does this image truly represent the original scene or was it digitally altered? These are just a few of the questions faced routinely by investigators and others in law enforcement.

A method to add information into images is through the *digital watermarking*: this is the process of embedding information, visible or invisible, into a digital signal or file, such as a image, in a way that is difficult to remove. Although digital watermarks have been proposed as a tool to verify the authenticity of images, the enormous majority of images captured today do not have one. This situation is likely to continue for the foreseeable future, so in the absence of their widespread adoption, we believe it is imperative to develop techniques that can generate definitive statements about the origin, veracity, and nature of digital images. The past few years have seen a growth of research in this area. Work in the field has focused mainly on solving two types of problems: image source verification and tamper detection. The aim of the former is to determine through what means a given image was generated (e.g., digital-camera, computer graphics, scanner, etc.) and then associate the images with a class of sources that have common characteristics, or match it to a specific source. Meanwhile, the goal

of tamper detection is to determine whether a given image has undergone any form of modification or processing since it was initially captured.

When the source of an image is identified as being a digital camera, the ability to identify its make and model requires an understanding of the image formation process. Although many of the details of the camera pipeline are considered proprietary information by manufacturers, the basic structure remains the same in all digital cameras and can be used to support the identification of the camera source, given an image. Indeed, the identification of the camera which generated an image can be critical in many trials.

Digital image forensic may allow the contrast with child pornography: the identification of the camera that took the photo can help to punish authors of the materials, determining whether the photo was altered can uncover if any detail has been hidden to prevent the identification of the location where the shooting occurred. Another critical point is determining if the image is real or virtual. Ultimately, defendants may claim that pornographic images are of "virtual" children, thus requiring the government to establish that the children shown in these digital images are real. In fact, if the image is virtual, since it is a 'drawing' and since there was no abuse of a minor, there is no offense.

Chapter 2

Digital imaging

As seen in the previous chapter, in digital forensics there is a growing need to perform analysis on digital images to achieve information useful in trials. Before to face the processes involved in the image analysis, it is necessary to introduce the image creation phase. This chapter briefly explains how is formed a digital image, and what are the possible problems that may make it less beautiful or mark an image.

Digital imaging is a process regarding the creation of digital images, typically from a physical scene. Traditional cameras capture images onto film while digital cameras use electronic chips known as a Charged Coupling Device (CCD) or Complementary metaloxidesemiconductor (CMOS). During the digital acquisition process the real image is converted into a series of electronic dots called pixels. Pixels is an acronym for *picture elements*. After the image is converted in pixels, or digitized, it is stored on a memory storage device which may be an hard drive or some sort of electronic storage device such as a memory stick. The pixels are stored, if necessary, in a compressed format to save storage space. Once each pixel is created, it is assigned a colour value, called tonal value, which can be black, white or shades of grey. Image sensors are essentially monochrome devices, i.e. they detect the amount of light incident on each pixel but cannot distinguish the colour of the incident light. To see the image with colors, the

pixels must be processed in order and with the use of the Bayer Array (a filter that passes either red, green, or blue light, as explained in details in sec. ADD SECTION REF).

These digital values are then stored on the cameras memory storage device. When these digital values are recalled by software, and displayed on a screen, they reproduce the image that was originally captured by the camera or digital input device. Indeed, before the actual image is recorded and transferred from the digital device, various causes can degrade the image therefore introducing *noise* in the images. The noise deteriorates imaging performance and determines the sensitivity of an image sensor. Therefore, the term *noise* in image may be defined as any signal variation that deteriorates an image.

In the following sections we explore in detail the images digital acquisition process, describing the issues in capturing the light and the problems which create noise in the images.

2.1 Capturing the light

As described above, there are two main types of sensor chips currently used in digital image capture devices: Bayer array charge coupled devices (CCD), and Bayer array complimentary metal on silicon (CMOS) chips. They vary in the arrangement and technology used to capture light from different portions of the image and how they then process that information.

All the digital light sensors commonly used in digital imaging devices are the same in that they comprise small tiles called pixels. All the pixels are part of a larger integrated circuit chip, referred to as a sensor chip. It is usually made on a silicon-based crystal. A pure silicon crystal is composed of silicon atoms in a regular array. Each silicon nucleus has a fixed location in a regular matrix and the electrons of each atom serve to bind the nuclei together. Silicon has a valence of four, and so each atom is bonded to

four surrounding atoms. In the absence of radiation, the crystal does not support the movement of electrical carriers, such as electrons.

In the case of silicon, when a photon of the right wavelength is absorbed, it will energize an electron to the point of being set free from the matrix. Once loose, the electron is relatively free to move around the crystal. The electron carries a negative electrical charge and is called a negative carrier. The movement of electricity in a copper wire is composed of the movement of electrons. In a silicon crystal, called a semiconductor, the release of an electron from its place in the matrix leaves behind a hole in the matrix. This is referred to as a hole and it, too, is somewhat free to move by a sequential substitution process. In the aggregate, the holes appear to be positive electrical carriers. If an electrical voltage is applied across the crystal, the negative carriers will move in one direction and the positive ones in the other. In summary, a silicon crystal is an insulator in the dark and a conductor in the light. The amount of charge that moves is equal to the number of photons absorbed and, therefore, proportional to the amount of incident light but not all the incident photons will be absorbed. The amount of charge released divided by the number of incident photons is referred to as the quantum efficiency of the crystal. The result is that the amount of charge left on the plates of the pixel when the exposure is stopped is linearly proportional to the illumination. The basic light gathering process starts out as an analog signal.

At very low light levels, some charge will flow just due to the fact that the chip is at some finite temperature. So there is a low limit of charge, which is variable and not due to light. This is noise. At very high light levels, more charge is created than the capacitor can hold, and the excess spills over to other circuit elements. This is saturation. At light levels below the noise limit and above the saturation limit, the photo sensor does not operate as a true indicator of light.

2.1.1 Bayer Array

The Bayer array is a Colour Filter Array (CFA) that comprises a repeating rectangular array of pixels, with each one covered by a filter as shown in Figure 2.1. A CFA is a mosaic of tiny color filters placed over the pixel sensors of an image sensor to capture color information. Each pixel is a light-sensitive portion of the main chip and is covered by a filter that passes either red, green, or blue light. Below the filters, each pixel is like all the others. The array pattern is what is referred to as the Bayer array. In this pattern half of all the pixels are green because this color is aligned most closely with the luminance of an image and is therefore the most significant contributor to the visual impression of sharpness. The green record must have the best resolution if the images are to appear sharp. The human eye also has an overabundance of green sensitive cones (i.e., the green receptors). The remaining half part of pixels is divided equally into red and blue pixels. The red pixels will be in one row interspersed by green, and the next row will have blue pixels interspersed with green. The same arrangement is maintained vertically. The result is that every blue pixel is surrounded by four green pixels and four red ones. Likewise, every red pixel is surrounded by four green pixels and four blue ones. All green pixels are diagonally surrounded by four other green pixels and at least two of each of the red and blue pixels.

The important properties of the Bayer array are that:

- each pixel captures light from only one of the three additive primary colors of red, green, and blue as shown in Figure 2.2;
- the highest sampling of the image is done in the green portion of the spectrum;
- there is close proximity of each pixel color to the other two colors.

The digital signal coming from a Bayer array chip is a string of numbers in which the value of each number is an indication of the amount of light that the particular pixel captured, and its location in the string is an indication of where that pixel was in

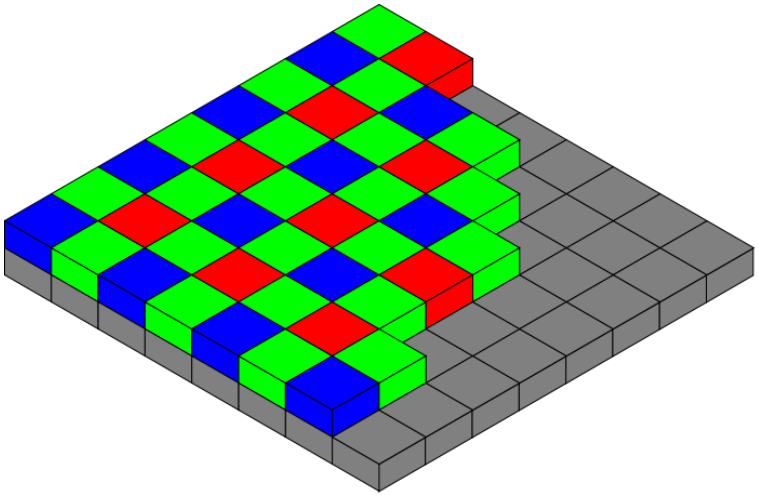

Figure 2.1: Bayer Array Pattern. Most digital still cameras utilize the Bayer array pattern, which places either a red, green, or blue filter on top of each pixel.

the original array. The color of the light which a pixel saw is inferred from its location in the original array. These data are just raw information: it is not a viewable image. The viewable image must be computed from this raw data and certain other indicators from the camera. In Figure 2.3, published in (4), is shown as the camera sees a physical scene.

All pixels have certain inherent properties that are important considerations in the design of a sensor chip. First of all, the larger the area of the pixel, the more sensitive it is. It is responding to photons per unit area, and the bigger the area, the more photons it will capture, all other things being equal. All pixels have a certain amount of dark current. The silicon is not supposed to generate carriers in the dark, but there is always some generation. This results in noise in the image in the form of a speckle pattern in the dark areas as we can see in the next section. The result is that chips with larger pixels have more signal and (usually) proportionately lower noise. Inevitably there will be a few defects in a chip. This can result in pixels that are too sensitive or not sensitive enough, compared to others on the chip. Testing and conditioning of the signal can minimize the impact of these defective pixels. Also, the process of passing the charge to

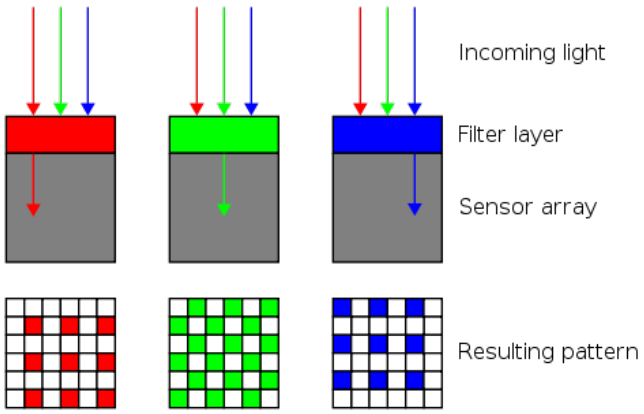

Figure 2.2: Bayer Array Pattern. Color discrimination with the colour filters on top of the sensor.

be read is not loss-less and totally consistent and produces noise. Again this noise will be most evident at low light levels. Silicon has a relatively low native sensitivity to the shorter wavelengths (blue) and a higher sensitivity to longer wavelengths (red). The result is that the dark current noise is more pronounced in the blue record. Fortunately people are not so sensitive to the blue record. But in order to keep a reasonable balance among the three records, the sensitivities of the green and red pixels are held back to be consistent with the blue ones. The final result is that the performance of the blue pixels tends to limit the overall performance of a chip.

Defects in a chip are due not only to the cost of the materials but are impacted by the incidence of defects on the wafer from which the chips are taken. If the wafer has, let's say, three defects, and a single chip is taken from that wafer, there are good chances that the chip will have at least two of those defects. If it was possible to work with chips that were half size on each side, then we could get four chips from the same wafer. It is now almost certain that one, and possibly two, of the chips will be defect free. The result is that smaller chips tend to have a higher yield-per-wafer and are therefore cheaper to produce. This is in addition to the reduced cost of the basic material. Given that larger pixels are desirable from a sensitivity and noise reduction

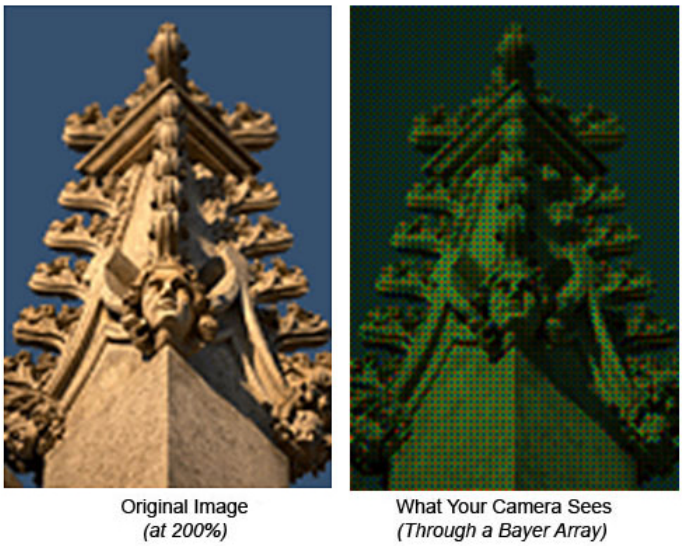

Figure 2.3: Bayer Array Pattern. As the camera sees a physical scene.

standpoint, but that smaller chips are cheaper to produce, there is a constant struggle between these two factors.

2.1.2 CCD and CMOS chips

CCD and CMOS chips have different structures but, performance-wise, they differ primarily in how the analog values from the pixels are converted to digital signals and how those signals are extracted from the chip. Except for battery life, most of the differences will not be noticeable to the user of a camera.

CCD chips operate by moving electrical charge, and the movement of electrical charge is, by definition, electrical current. In other words, CCD chips are current-based devices and consume significantly more power, so in comparison to CMOS devices, the batteries will last a bit less than those in a CMOS camera. CCD have a single ADC (a built-in Analogy-to-Digital Converter which turns the stored electrical value into a digital value) and there is only a minimum of circuitry on the image-sensing area, instead CMOS cameras have a large amount of control and access circuitry attached to each pixel, and this takes up valuable space on the sensory portion of the sensor chip.

The result is that CCD chips use a larger proportion of the area of the light-sensitive portion of the chip for pixels. This means that they tend to have lower noise and higher sensitivity. Also, with the CCD array, the adjacent sensitive areas of the pixels are closer to each other. This means that there are fewer gaps in the sampling of the image.

2.2 Image noise

Before the actual image is recorded and transferred from the digital device, various noise sources degrade the image. Noise deteriorates imaging performance and determines the sensitivity of an image sensor. Therefore, the term *noise* in image may be defined as any signal variation that deteriorates an image. Some of these noise sources are temporal, some of these are spatial and others are a combination of these. An image sensor reproduces two-dimensional image (spatial) information.

Noise appearing in a reproduced image, which is "fixed" at certain spatial positions, is referred to as fixed-pattern noise (FPN). Because it is fixed in space, FPN at dark can be removed, in principle, by signal processing. Noise fluctuating over time is referred to as "random" or "temporal" noise.

2.2.1 Fixed-pattern noise

The primary FPN component in a CCD image sensor is dark current non-uniformity. Although it is barely noticeable in normal modes of operation, it can be seen in images that have long exposure times or that were taken at high temperatures. If the dark current of each pixel is not uniform over the whole pixel array, the non-uniformity is seen as FPN because the correlated double sampling (CDS) cannot remove this noise component. In CMOS image sensors, the main sources of FPN are dark current non-uniformity and performance variations of an active transistor inside a pixel.

Some of the variations due to dark current are somewhat systematic: the spatial

pattern of these variations remains constant. Because of fabrication and material properties, this *fixed pattern noise* (FPN) is a flat field uncertainty due to device response when the sensor is not illuminated. Crystal defects, impurities and dislocations present in the silicon may contribute to the size of the fixed pattern noise. In CMOS image sensors additional sources are present, and can be thought of as composed of a column component (shared between all pixels in a certain column) and an individual pixel component. As FPN is added to all frames or images produced by a sensor, and is independent of the illumination, it can be easily removed by subtracting a dark frame from the image.

A source somewhat similar in characteristics to FPN is PRNU (Pixel Response Non-Uniformity), the variation in pixel response when the sensor is illuminated. This variation comes e.g. from non-uniform sizes of the active area where photons can be absorbed. This is a linear effect. For example, when the size of the active area is increased with a factor x, the number of photons detected will also increase with factor x. This illustrates the multiplicative characteristic of the PRNU: when the illumination increases, the effect of this source increases as well. Another possibility is the presence of non-uniform potential wells giving a varying spectral response. Therefore, the PRNU is also wavelength dependent. The multiplicative nature of the PRNU makes it more difficult to remove this type of non- uniformity, as simply subtracting a frame does not take this illumination dependent nature into account. In principle it is possible to remove the PRNU, or even add the pattern of a different camera. It is also possible to reduce the PRNU inside the camera by a form of non-uniformity correction.

FPN together with PRNU form the pattern noise and is always present, though in varying amount due to the varying illumination between successive frames.

2.2.2 Temporal noise

Temporal noise is a random variation in the signal (which is assumed to be constant), that fluctuates over time. Three types of fundamental temporal noise mechanisms exist in optical and electronic systems: shot noise, termal noise, and flicker noise. All of these are observed in CCD and CMOS image sensors.

Temporal noise in image sensors is mainly due the *shot noise* that is inherent to the nature of light and to a lesser extent to the *thermal noise* due the thermal generation of charge carriers in the silicon substrate of the image sensor. As the camera has no way of differentiating the signal charge from the spurious electrons generated, these unwanted electrons are added to the output and represent a noise source. *Flicker noise* is also a temporal noise source, in which charges are trapped in surface states and subsequently released after some time in the charge to voltage amplifier. In CMOS active pixel sensors additional sources are present due the various transistors integrated on each pixel. As this temporal noise is a purely statistical phenomenon, averaging multiple frames will reduce the amount of temporal noise.

2.2.3 Other types of noise

There are also noise sources that do not find their origin on the image sensor but are added further down the pipeline, i.e. when the digital signal is processed. The most obvious source of this type of noise is the *quantisation noise* introduced when the analogue information from the sensor (the potential change detected for each pixel) is digitised in the analogue-to-digital converter. Another effect that occurs in the processing stage is the demosaicing of the signal. The demosaicing is the process where the full color image is reconstructed by interpolation from the incomplete color samples output obtained with the CFA on the image sensor. This interpolation gives small but detectable offsets, and can be seen as a noise source. Also, dust present on the lens may contribute to the pattern noise, as well as possible optical interference in the lens

system.

Chapter 3

Camera Identification

As described in the chapter *Digital Image* every digital picture contains a random component of noise as well as a deterministic component, the *pattern noise*, that depends on the sensor used to shoot the picture. The pattern noise is "very" similar within all the pictures taken by the same sensor and can be used to identify the camera.

The problem of digital camera identification concerns with the identification of the camera that has been used to generate a digital picture, by examining the pattern noise in the picture. This technique is called *source camera identification* and should not be confused with the more general *digital camera model identification* problem, in which there is only interest in establishing which camera model has been used to take a certain picture. In this chapter we present the generic problem of the source camera identification and a brief survey of the approaches existing in literature, with particular attention to the Lukáš *et al.* approach, which currently results as one of the most effective. However, in some case this method presents difficult to correctly associate images to the rigth source camera. Consequently, our aim was to improve this method by adopting a SVM (Support Vector Machine) classifier, technique supporting classification and regression analysis based on supervised machine learning.

3.1 Source Camera Identification

As a matter of fact different imaging devices have different characteristics due to the use of different physics apparatus, different image processing, and different parameters applied inside the imaging devices, etc. Thus it would lead to different patterns of the output images, so these patterns can be used as inherent *fingerprints* of the imaging devices to identify the source of the image.

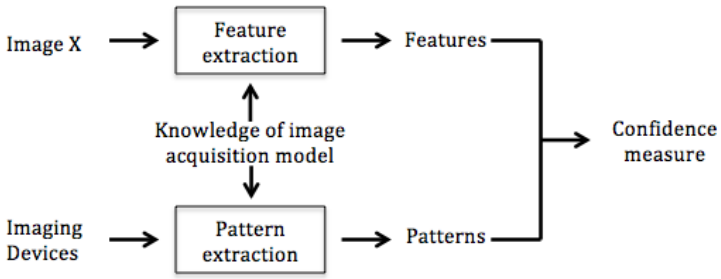

Figure 3.1: Generic image source identification

In the generic problem to match an image with its source camera, we suppose to have an image to be detected and a set of candidates of source camera, and the aim is to associate the image with the right source camera. This scheme is illustrated in Figure 3.1. The image X is to be detected, and may come from one of the candidate imaging devices. The identification process is as follows. Firstly, the features from X and the patterns from the imaging devices are extracted mainly using the knowledge of image acquisition model. Then the similarities between these patterns and the characteristic features are measured. Lastly, a confidence measure for each imaging device to identify the source of the image X is given. In literature, this generic scheme has been followed by several approaches, and a brief survey is presented in the next section.

3.2 State of the art

Up until now, three main approaches were proposed in literature to deal with the source camera identification problem. These approaches differ in the type of pattern noise used.

- The first approach uses the *Photo-Response Non-Uniformity* (PRNU) noise, i.e., the noise produced by the sensor due to the inhomogeneity of the silicon wafers used to build it. Lukáš *et al.* (29) and Chen *et al.* (11) proposed two methods to identify the source camera based on the PRNU. These techniques can be used to isolate and extract the noise pattern from a set of pictures taken with the same camera, using this pattern to match or not the cameras with the photos under investigation. Their results show that these methods have high detection rates. Goljan *et al* in (18) used a refinement of the method of Chen to run a digital camera identification test on a massive database of digital pictures downloaded from the Internet.

- The second approach uses the lens radial distortion that causes straight lines to appear as curved lines on the output images. Choi *et al.* have tried this method in (12). They discovered that this method failed to measure the radial distortion except when there are explicit straight lines in the picture to be processed.

- The last approach relies on the *Color Filter Array* (CFA) interpolation, which is a technique used by digital cameras after a picture has been taken in order to determine the colors of the scene. This technique produces small non-uniform color zones that can be seen as a noise source. Every camera has its own interpolation algorithms and produces a small degree of noise that, generally, changes slightly from one camera to another. Bayram *et al.* in (2) explored the CFA interpolation process to determine the correlation structure present in each color band which can be used for image classification. In this direction, Kharrazi *et al.* in (25) and

Long and Huang in (28) proposed two methods. The first method identifies a set of image features that can be used to uniquely classify a camera model. The accuracy of this method decreases as the number of cameras increases. The second method obtains a coefficient matrix from a quadratic pixel correlation model where spatially periodic inter-pixel correlation follows a quadratic form. The results seem to suggest that these two methods work better for the problem of camera model identification rather than for that of source camera identification.

3.3 Lukáš's method

We have focused our attention on the approach proposed by Lukáš *et al.* since it is one of the most effective method, as well as inexpensive in terms of hardware resources unlike other similar methods such as the one proposed by Goljan in (18).

The approach by Lukáš *et al.* works in two stages. In the first stage, the PRNU associated with a CCD sensor is determined by analyzing a batch of images taken with the sensor. In the second stage, given a picture, the procedure evaluates the correlation between the noise in the picture and the pattern noise evaluated in the previous stage in order to distinguish whether the picture has been taken using that CCD sensor or not.

The extraction of the PRNU from an image is performed by denoising the image using a wavelet-based algorithm. The denoised image is subtracted from the original image giving as output a new image containing several components: the CCD sensor noise, the random noise and various contributions from the image signal. Thus, in order to eliminate the random component of the noise, the denoising procedure is applied to a set of images (captured by the same camera) and the corresponding noise residues are averaged to obtain the reference pattern of a given digital camera.

In practice the PRNU acts as a device signature and it is used as the unique characteristic. PRNU presents itself as a visually invisible pattern that is present in all images

or videos a camera produces. However, the degree in which this pattern is available depends on the illumination; in well illuminated segments of an image the PRNU can be more reliably estimated than in segments with low intensities. With the use of filters, these characteristic patterns can be extracted from images, or from individual frames from videos.

Afterwards, to determine whether a given image is captured by a digital camera, the noise pattern extracted from the given image is correlated with the reference pattern of the camera. If the correlation value exceeds a pre-determined threshold, then the image was taken with that camera. In order to estimate the accuracy of the method as well as to compute the thresholds, the Neyman-Pearson criterion was used, specifying a bound on the FAR.

3.3.1 PRNU Filter

The PRNU noise is induced by intrinsic inhomogeneities over the silicon wafer and imperfections generated during sensor manufacturing process of CCD/CMOSs. The PRNU is used as sensor fingerprint and it is commonly employed to solve the problem of digital camera sensor identification. The extraction of PRNU noise happens through a denoise filtering operation from a set of digital images taken by a camera.

Figure 3.2: PRNU sample. Real image taken with camera

There are many image denoising filters. The purpose of the denoising filter is to obtain an approximation to the pixel non-uniformity noise and to remove the influence of the image scene. A general-purpose denoising filter is described in (26). This filter

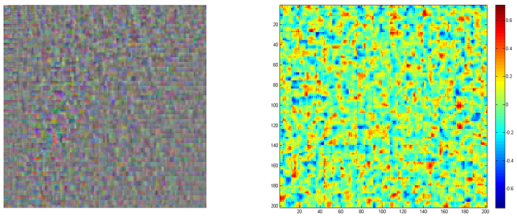

Figure 3.3: PRNU sample. (a) PRNU pattern extracted from test image. (b) PRNU extracted pattern (variantions in greylevels in colour)

extract from the image a Gaussian noise with known variance (an input parameter to the filter), based on the assumption that, in wavelet domain, the image and the noise form on addictive mixture of a non-stationery Gaussian signal with a known variance. Experiments show that the value 5 for the variance gives the best overall performance across all devices.

3.4 SVM Classification

Support vector machines (SVMs) are a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis. Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value, the supervisory signal. A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier (if the output is discrete, see classification) or a regression function (if the output is continuous, see regression). The inferred function should predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data to unseen situations in a “reasonable” way.

As describe above, SVM are a useful technique for data classification. A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one "target value" (i.e. the class labels) and several "attributes" (i.e. the features or observed variables). The goal of SVM is to produce a model, based on the training data, which predicts the target values of the test data given only the test data attributes.

SVM requires that each data instance is represented as a vector of real numbers. Hence, if there are categorical attributes, we first have to convert them into numeric data. A best practice to this is to use m numbers to represent an m-category attribute. Only one of the m numbers is one, and others are zero. For example, a three-category attribute such as red, green, blue can be represented as (0,0,1), (0,1,0), and (1,0,0). If the number of values in an attribute is not too large, this coding might be more stable than using a single number. Scaling before applying SVM is very important. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors large attribute values might cause numerical problems. Of course the same method to scale both training and testing data must be used.

3.4.1 Using SVM to identify the camera

All the tests confirmed that the pixel non-uniformity noise could be successfully used to identify the image source but for some specific camera models we got a sort of pattern clash and many images produced low correlation values with more than one reference pattern previously extracted from the cameras under investigation including the true source. In these cases the values were also too close to the rejection threshold. This has been considered an important issue to adopt the Lukáš *et al.* approach for digital image forensics. Moreover the process to define the acceptance threshold

based on the Neyman- Pearson approach, which has been originally employed to define the acceptance threshold t minimizing the false reject rate (FRR), resulted too time consuming and not enough flexible because the necessary accuracy can be reached only if the system can learn from the analysis of a considerable number of images from a known source and when the camera set or the input image set change all the threshold t must be recomputed.

So we decide to use SVM to associate an image to a camera. We first define a set S of candidate digital cameras, than we extract the sensor noise reference pattern for each camera. Then we take a small set of randomly chosen pictures for each camera. For each image we calculate the correlation factor against the reference pattern of each camera. Finally we use these values as training set for a SVM classifier (Support Vector Machine) with a number of input attributes equal to the cardinality of the set S. Similarly we defined a test set to verify the result of the training. In fact we recalculate the correlation with all the reference patterns and we give the resulting values as input to the SVM which can provide us with the source camera identification. We have extensively tested the approach we propose and, as shown in the chapter *Experimental Analysis*, it led us to a double improvement:

1. on one hand our approach improved the accuracy of the final result especially in the case of cameras with sensor noise patterns which produces low correlation values or simply too close to be considered distinct. In facts, the SVM can identify the source discriminating these values considering the correlation values produced by the patterns of the other cameras of the set S instead of comparing each correlation value against a single predefined threshold value.

2. on the other hand our approach produced a considerable speed up of the whole process for source camera identification, reducing the amount of the images necessary to train the system establishing the threshold of acceptance. This becomes even more meaningful if is considered that during the investigation often we had

to work with a set of hundred candidate cameras, and many of these belong to the same producer/model.

Chapter 4

Experimental analysis

In this section will be presented the tests conducted on two source camera identification methods. The first is our implementation of the Lukáš *et al.* technique and the second the technique we have developed where SVM is used to associate images to the right camera. The Lukáš *et al.* technique has been implemented by us as it is not publicly available and the behaviour of our implementation is in line with their results (29). Both the methods needs an images dataset, which was split in two sets: one to compute the reference patterns (or fingerprint) of the camera, and one to check the effectiveness of the techniques. After the creation of an images dataset, several tests will be conducted with both of methods in order to verify the correctness and to identify which performs better.

4.1 Implementation of methods details

In the next subsections will be provided some information about the implementation of the methods.

4.1.1 Method proposed by Lukáš *et al.*

In the Lukáš *et al.*ś method the device identification works through the extraction of the noise pattern from the image to be detected, the extraction of the reference noise patterns for each source camera candidate and therefore it performs the comparison among the image noise pattern and the reference patterns:

- The noise pattern is extracted from the questioned image and denoted as the *natural pattern*;
- *Reference patterns* are generated for each source camera candidate by making flatfield images: images that do not contain scene content and have an (approximate) uniform illumination. The PRNU patterns are then extracted from each of these flatfield images, to generate a reference pattern for each camera.
- If the natural pattern and one of the reference patterns have a high degree of similarity, it is an indication that the natural image originates from this camera. Of course, the natural pattern and the reference patterns need to have the same size.

We have implemented the method proposed by Lukáš *et al.* by using the Matlab software (31). This software was used due to it being efficient as well as providing several pre-implemented components (e.g. wavelets functions) which are useful in the implementation of the various identification techniques.

Concerning the implementation, the main element of the method proposed by Lukáš *et al.* is the PNRU filter used in the extraction of the noise patterns. This filter simulates the behaviour of the Wiener filter in the wavelet domain and it has been suggested in (26). The Wiener filter is based on a statistical approach and aims to filter the noise of an image.

There are several families of wavelets, each one suitable for different applications, differing in the number of coefficients they use. In the early stages of the tests, several

combinations were tried, with the optimal choice being 4-levels and 8-levels Daubechies wavelets.

4.1.2 Method based on SVM

In this method device identification is based on PRNU and noise pattern extraction, as the method of Lukáš *et al.* but we use a different policy to associate an image to a camera. The association is not based on a threshold but on a SVM classifier. The method works as follows:

- *Reference Patterns* (RP) are generated by making flatfield images likes in the method of Lukáš *et al.* ;
- The noise pattern is extracted from a number of, randomly chosen, images for each camera of the dataset. For each image is computed the correlation factor against the reference pattern of each camera and these constitutes the pattern to be used for the training of the SVM. Finally these values were used as training set for the classifier. The training set is composed in the following manner. Let N the number of cameras and K the number of pictures chosen for the training for each camera. Let M_j the camera number j ($1 <= j <= N$). Let P_{i,M_j} the picture number i taken with the camera number j ($1 <= i <= K, 1 <= j <= N$). Let $RP(x)$ the function that computes the reference pattern on camera x, $RN(y)$ the residual noise on the pictures y and $Cor(x, y)$ the correlation factor between the images x and y. The training set file is composed, except the emphasized text used only for description, as indicated in the following table. For clearness of the table indicate $C(M_i, P_{j,M_p})$ instead of $Cor(RP(M_i, RN(P_{j,M_p})$
- Similarly a test set was defined to verify the result of training.

There are various software packages implementing Support Vector Machines. In our experiments we use LIBSVM (10). LIBSVM is an integrated software for classifica-

class label	$RP(1)$	$RP(2)$	...	$RP(N)$
1	$1 : C(M_1, P_{1,M_1})$	$2 : C(M_2, P_{1,M_1})$	...	$N : C(M_N, F_{1,M_1})$
1	$1 : C(M_1, P_{1,M_1})$	$2 : C(M_2, P_{1,M_1})$	...	$N : C(M_N, F_{1,M_1})$
		...		
1	$1 : C(M_1, P_{1,M_1})$	$2 : C(M_2, P_{1,M_1})$	...	$N : C(M_N, F_{1,M_1})$
2	$1 : C(M_1, P_{1,M_2})$	$2 : C(M_2, P_{1,M_2})$	...	$N : C(M_N, F_{1,M_2})$
2	$1 : C(M_1, P_{1,M_2})$	$2 : C(M_2, P_{1,M_2})$	...	$N : C(M_N, F_{1,M_2})$
		...		
2	$1 : C(M_1, P_{1,M_2})$	$2 : C(M_2, P_{1,M_2})$	...	$N : C(M_N, F_{1,M_2})$
		...		
N	$1 : C(M_1, P_{1,M_N})$	$2 : C(M_2, P_{1,M_N})$	...	$N : C(M_N, F_{1,M_N})$
N	$1 : C(M_1, P_{1,M_N})$	$2 : C(M_2, P_{1,M_N})$	...	$N : C(M_N, F_{1,M_N})$
		...		
N	$1 : C(M_1, P_{1,M_N})$	$2 : C(M_2, P_{1,M_N})$	...	$N : C(M_N, F_{1,M_N})$

tion, regression, and distribution estimation. The classification methods supported by LIBSVM include C-SVC and nu-SVC, and the former one was selected for these work.

4.2 Experimental settings

Experimental settings describe the image dataset used and how this is arranged. Seven different camera models were considered, resulting in eight cameras (as shown in Table 4.1). In order to stress the identification methods as well as cover a wider range of hardware, cameras belonging to different market sectors and different manufactures were chosen. Looking at Table 4.1, cameras with ID 1 and 2 were chosen because they have the same image sensor size as well as the same CMOS sensor (6). Cameras with

Table 4.1: Cameras used in the tests.

ID	Model	Sensor	Image size
1	Canon EOS 400D	CMOS	3888x2592
2	Canon EOS 1000D	CMOS	3888x2592
3	Canon PowerShot A400 instance A	CCD	2048x1536
4	Canon PowerShot A400 instance B	CCD	2048x1536
5	Panasonic Lumix DMC-FZ20	CCD	2048x1536
6	Panasonic Lumix DMC-FS5	CCD	3648x2736
7	Kodak EasyShare CX 7530	CCD	2560x1920
8	HP PhotoSmart E327	CCD	2560x1920

ID 3 and 4 share the same brand and model. The other four cameras are a mix of common cameras.

For each camera model $c \in C = \{1, 2, \ldots, 8\}$ two sets of images were collected: the *Images for Reference Pattern* (IRP) and the *Images for Testing* (IT). IRP_c / IT_c denotes the IRP / IT sets for the camera c. The IRP_c set is composed of 128 images collected by taking pictures of a uniform white surface. The images were taken on a tripod, with no flash, auto-focus, no-zoom, best JPEG compression quality, and with all the other options set to their default values. The IT_c set is made up of 180 images portraying different types of subjects. In this case, the images were taken using different types of settings, with the exception of the JPEG compression quality as well as the image size, which were always set to maximum. In Figure 4.1 there are a sample of images collected for IT set.

As the SVM classifier needs a set of test images and a set of training images, we split, for each camera c the dataset IT_c into two new sets: $\mathrm{IT}_{tra,c}$ and $\mathrm{IT}_{tes,c}$. The $\mathrm{IT}_{tra,c}$ set is composed of 30 images chosen randomly from the origin dataset IT_c, while the remaining part of the IT_c set was used to populate the $\mathrm{IT}_{tes,c}$ set.

The effectiveness of the identification was measured by counting the number of pictures erroneously rejected by the identification techniques over the total number of

Figure 4.1: Some pictures of the image dataset.

pictures taken with a certain camera. Moreover, in all the tests, the decision thresholds for the method of Lukáš *et al.* were set in such a way to keep to 0 the total number of pictures erroneously classified as taken with a certain camera. All the tests were run on a server equipped with two 4-core Intel Xeon X7350 processors at 2.93GHz and using the Linux Ubuntu operating system.

4.3 Experimental explanation

All the following tests have been conducted on a data set composed of images that have been knowingly modified in order to test the effectiveness of the identification tecniques by Lukáš *et al.* and of SVM classifier.

Three kinds of experiments have been planned. In the first experiment, the effectiveness of the identification methods when applied to the camera identification for unmodified digital pictures was assessed. The second experiment and the third experiment regards, respectively, the identification of images processed with some software of photo editing and of images published on online social network and photo sharing sites.

4.3.1 How to correlate images of different size?

A preliminary problem to be faced in the computation of the correlation between two images is to ensure that the two images t (i.e., the image reference pattern and the image to be identified) have the same size. This condition can be easily met in three different ways:

- Extract from both images two sub-images of the same size (**Sub**). In our case, extract two images originating at point $(0, 0)$ and having size 512x512;
- Crop the larger image to match the smaller image (**Crop**);
- Resize the larger image to match the smaller image (**Resize**).

The original method proposed by Lukáš *et al.* uses the **Crop** approach. In this study, it was decided to also test the other two approaches in order to determine which one performs better.

4.3.2 Photo editing

This experiment was intended to assess the resilience of the identification methods when used for classifying pictures that have been subjected to some sort of pre-processing. This experiment were organized in two phases. In the first phase, the original set of pictures were initially pre-processed using several types of image-processing functions, with identification process then being repeated, using the decision thresholds and the training model established during the first experiment according to non pre-processed images. In an second phase, the previous tests with pre-processed images were repeated, using the decision thresholds and a new training model that have been recalculated from pre-processed images.

The experiment was organized by first applying six different commonly-used image processing operations to the data set described in Experimental settings. Then, the identification method on the resulting data sets was applied, using, for each camera, the same reference pattern and decision threshold or training model established in the experiment that establish the best approach to correlate two images of different size. Finally, the resulting classification was compared with the results of the classification on the original (i.e., not pre-processed) pictures. The operations that were considered in the experiment, as implemented by the Adobe Photoshop software (1), are:

- **Auto Level Adjustment** (*ALA*): this function automatically corrects the highlights and shadows in a picture and adjusts the tones so that the lowest level in the picture is completely black and the brightest white is full white. Auto Levels tunes each color channel individually, and this may remove or introduce color casts;

- **Auto Contrast** (*ACS*): this function adjusts the overall contrast and mixture of colors in an image, without introducing or removing color casts, and permits to create a more accurate tonal and color-correction;

- **Auto Color** (*ACO*): this function adjusts contrast and color of an image by neutralizing the midtones and clipping the white and black pixels;

- **Resizing** (*R75*, *R50*, *R25*): this operation rescales the image to match a smaller size; the interpolation algorithm is *bi-cubic* which produces noticeably sharper images than other methods such as *bilinear* or *nearest neighbour*, and it is a good balance between processing time and output quality. The images were processed with this operation by changing the scale factor. Pictures with the image size of 75%, 50% and 25% of its original sizes were obtained.

4.3.3 Online photo sharing

All the previous experiments have been conducted on unmodified pictures or on pictures that have been intentionally modified by the end-user, using photo-editing tools. In the real world, it is quite usual to upload digital pictures on online photo sharing (OPS) or online social networks (OSN) websites, without any prior modification. This could lead to the wrong conclusion that the pictures found on these sites retain the same properties of their original counterparts and, so, that can be used for the digital identification process. Instead, OSNs and OPSs websites usually process uploaded pictures in order to reduce their size and to speed-up their handling. The arising question is: does this pre-processings puts at risk the effectiveness of the Lukáš *et al.* and SVM identification technique when applied to pictures retrieved from one of these sites?

This preliminary test was aimed at determining which OSNs and OPSs modify pictures uploaded by users. This has been done, first of all, by choosing a set of OSNs and OPSs according to their popularity. Then, several sets of pictures have been uploaded and downloaded from all these sites. The downloaded pictures have been analyzed in order to understand if, and how, they have been modified.

In Table 4.2 the results of one of these experiments, carried out with a sample picture of 3.888x2.592 pixels and size of 2.275 kilobytes, are presented. For each site,

it has been checked if the picture was modified or not and, in the former case, it has been measured the size and the resolution of the modified picture.

These experiments show that only the following OSNs and OPSs, among the considered ones, process and modify uploaded pictures.

Table 4.2: Modifications performed by several OSN / OPS sites on a target image of resolution 3.888x2.592 pixels and size 2.275 kilobytes.

Service	Modified	Modified image resolution	Modified image size
Facebook	Yes	720x480	53 Kb
Flickr	No	3.888x2.592	2.275 Kb
MySpace	Yes	600x399	33 Kb
PhotoBucket	Yes	1.023x682	131 Kb
Picasa	No	3.888x2.592	2.275 Kb
Twitpic	No	3.888x2.592	2.275 Kb

Facebook (FAC) is currently the most used OSN in the world, with more than 500 million active users. It offers a relatively simple support for uploading and sharing photos. No limit is apparently put on the size of the pictures that can be uploaded.

Currently, service administrators do not disclose any information about the way images are processed and stored on their servers. However the experiments revealed a strong compression, via downsampling, of all the pictures uploaded with a standard resolution of 720 pixels on the long edge. This is evidently done in order to cope with the huge amount of pictures uploaded daily.

Recently, service administrators have announced an upgrade on the maximum size of the images stored in the Facebook database. According to this new setting, it will be possible to upload also high-resolution images, with a maximum size of 2.048 pixels on the long edge.

Photobucket (PHB) is one of the most popular OPS with a massive audience of more than 23 million monthly unique users in the U.S., and over four million images uploaded per day from the web and smartphones (34). Photobucket offers a simple support for uploading and dowloading and sharing photos. Like in the Facebook case, no information is disclosed about the way pictures are stored on their servers. However, the experiments revealed a compression process, albeit less aggressive than the one used by Facebook.

MySpace (MSP) is a OSN where users can share music, videos and pictures. No limit is put on the number and on the size of the uploadable pictures. However, even in this case, the experiments revealed a strong compression of the uploaded images, both in terms of size and downsampling.

As for the second experiment, also this were organized in two phases. In the first phase, the Lukáš *et al.* and SVM identification techniques has been experimented by applying it on pictures previously uploaded on each of these websites. This experimentation has been first conducted by using the same decision threshold and training model computed in experiment 1 of the previous section (see 4.3.1).

In the second phase, the Lukáš *et al.* and SVM identification techniques has been experimented again on the same set of pictures of the previous test by using, this time, thresholds computed by means of images stored and retrieved from the considered OSNs/OPSs.

4.4 Experimental analysis of the Lukáš identification technique

In this section we are going to present the results of experiments described in 4.3 by using the Lukáš identification technique.

4.4.1 Experiment 1 - How to correlate images of different size?

According to the results, presented in Table 4.3 (Lukáš *et al.*) the best approach seems to be **Resize** while the worst is **Sub**. The reason for such a bad performance is likely to be due to the elimination of a large part of the original image, when processing large pictures. The average resolution of the pictures used in the tests is near 2560x1920. As a consequence of this, the cropped image, whose size is fixed to 512x512, retains only the 5% of the original image as well as its signature, and thus is subject to a worse correlation. In all the remaining experiments presented in this thesis, when needed, the **Resize** technique will be used.

Table 4.3: Decision thresholds, FRR and number of images rejected on the red channel for the tests Sub, Crop and Resize.

	Sub		**Crop**		**Resize**	
ID	Decision threshold	Images rejected (FRR)	Decision threshold	Images rejected (FRR)	Decision threshold	Images rejected (FRR)
1	0,021	20 (0,111)	0,008	—	0,008	—
2	0,021	29 (0,161)	0,007	—	0,007	—
3	0,024	18 (0,1)	0,018	9 (0,05)	0,018	9 (0,05)
4	0,024	6 (0,033)	0,025	—	0,025	—
5	0,052	1 (0,005)	0,046	1 (0,005)	0,035	—
6	0,026	5 (0,027)	0,018	—	0,018	—
7	0,013	156 (0,866)	0,003	138 (0,766)	0,003	2 (0,011)
8	0,037	5 (0,027)	0,022	3 (0,016)	0,014	—
Tot.		240		151		11

Figure 4.2 presents the scatter plot of the correlations between all the images of the data set and the reference pattern of the Canon PowerShot A400 instance A (the camera with ID 3).

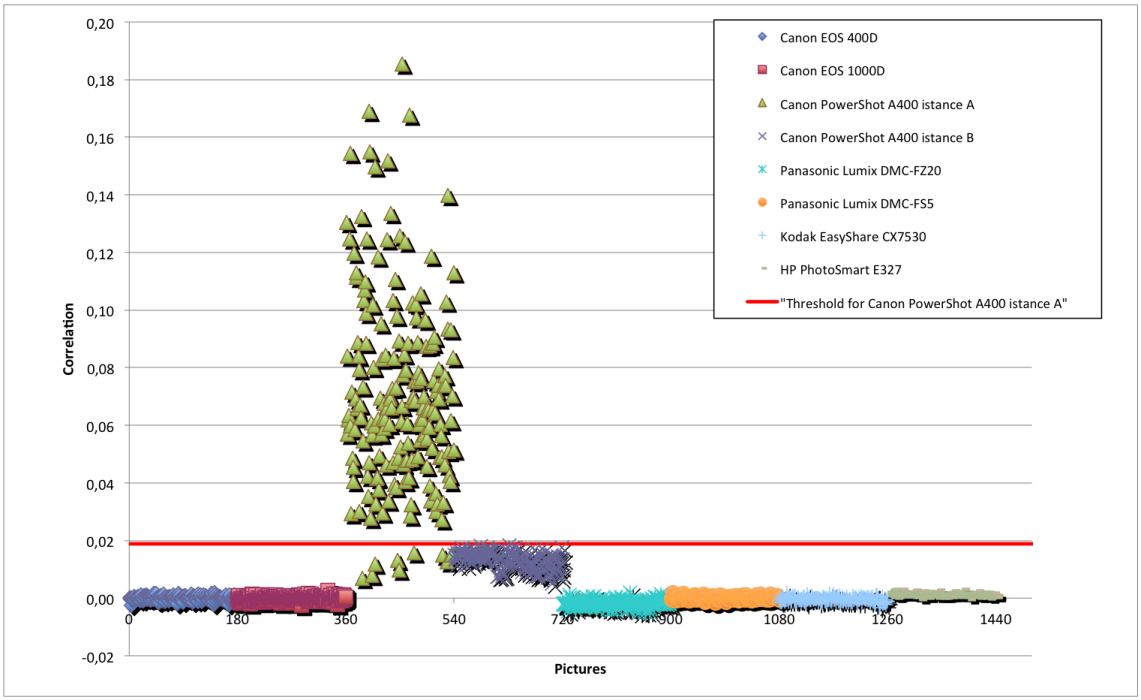

Figure 4.2: Scatter plot of the correlations between all the images of the data set and the Image Reference Pattern of the camera with ID 3.

4.4.2 Experiment 2 - Photo editing

As described in section 4.3.2 the experiment was conducted in two phases. In the first one, we have used the thresholds computed in the experiment section 4.4.1 and in the second one the thresholds are recalculated on the processed images.

Confidence measure of experiment 1

The results of this experiment, presented in Table 4.4, are noteworthy. A small increase in the number of erroneously rejected images can be observed when considering the pictures processed with the ALA, ACS and ACO operations. This increase is much more significant when considering the resized images. Here, the number of rejected images is high and grows linearly with the resize factor.

By examining in details these results, it is worth noting that there are some camera

	Operation						
ID	**No**	**ALA**	**ACS**	**ACO**	**R75**	**R50**	**R25**
1	—	—	—	—	—	—	2
2	—	4	4	3	2	2	14
3	9	11	11	11	11	15	49
4	—	3	3	3	3	7	51
5	—	1	1	1	—	1	108
6	—	7	7	7	10	12	97
7	2	2	2	2	2	2	2
8	—	1	1	1	2	2	40
Tot.	11	29	29	28	30	41	363

Table 4.4: Number of images rejected on manipulating pictures with thresholds computed in Experiment 1 (Resize, 4.4.1).

models where the identification method performs very poorly when used with resized images. It is the case of models 3, 4, 6, and, especially, model 5. This seems to suggest either that the resize operation may have a very strong influence on the correlation between the picture and the reference pattern noise, and that this influence may vary greatly according to the camera being used, even for different cameras of the same model. Moreover, it can ben noted that if the decision thresholds are chosen using, as a reference, pictures that have not been previously pre-processed, the identification method may likely fail.

Confidence measure recalculated

In the previous test, it can be observed that when trying to classify pre-processed pictures using a classifier that has been tuned for unmodified pictures, the identification method by Lukáš *et al.* may fail, in some cases, with a very high probability. These failures are mostly due to the alteration of the pattern noise existing in a processed picture. This alteration implies a smaller correlation with the reference pattern noise. A natural solution to this problem consists in lowering the decision threshold used during

the classification, so as to also correctly identify pictures with smaller correlations.

	ALA		ACS		ACO	
ID	Decision threshold	Images rejected (FRR)	Decision threshold	Images rejected (FRR)	Decision threshold	Images rejected (FRR)
1	0,008	—	0,008	—	0,009	—
2	0,004	2 (0,011)	0,004	2 (0,011)	0,005	2 (0,011)
3	0,022	11 (0,061)	0,019	11 (0,061)	0,019	11 (0,061)
4	0,019	2 (0,011)	0,02	2 (0,011)	0,02	2 (0,011)
5	0,035	—	0,035	—	0,0352	—
6	0,001	7 (0,038)	0,003	7 (0,039)	0,0019	7 (0,039)
7	0,003	2 (0,011)	0,003	2 (0,011)	0,003	2 (0,011)
8	0,014	—	0,014	—	0,014	—

Table 4.5: Decision thresholds, FRR and number of images rejected on the red channel for the tests ALA, ACS and ACO.

	R75		R50		R25	
ID	Decision threshold	Images rejected (FRR)	Decision threshold	Images rejected (FRR)	Decision threshold	Images rejected (FRR)
1	0,011	—	0,01	—	0,009	2 (0,011)
2	0,004	2 (0,011)	-0,001	—	0,0067	9 (0,05)
3	0,011	11 (0,061)	-0,002	—	0,009	11 (0,061)
4	0,013	2 (0,011)	0,009	2 (0,0111)	0,007	2 (0,011)
5	0,037	—	0,033	—	0,017	—
6	0,002	7 (0,039)	0,003	7 (0,0389)	0,005	13 (0,072)
7	0,004	2 (0,011)	0,006	2 (0,0111)	0,009	2 (0,011)
8	0,013	—	0,009	—	0,007	5 (0,028)

Table 4.6: Decision thresholds, FRR and number of images rejected on the red channel for the tests R75, R50 and R25.

The results, documented in Table 4.5, show a significant improvement in the quality of the classification, with respect to the previous test. In this case, it has been possible to obtain FRR rates which are very similar to those experienced with the first test. However, such a result comes at a cost. The new decision thresholds are, in some cases,

much lower than the original ones. For example, the decision threshold relating to camera 5 had to be lowered to more than 90% of its original value, thus raising the possibility of wrong classifications on larger data sets.

The same behaviour can be noted when using the R75, R50, and R25 operations. As shown in Table 4.6, even for these operations, the thresholds change without any correlation with the percentage of resize. In other words, what should have been expected with this test is that reducing the image size would decrease the correlation index. This happens only in some cases like for example in the case with camera ID 8. A graphical representation of the updated decision thresholds is available in Figure 4.3).

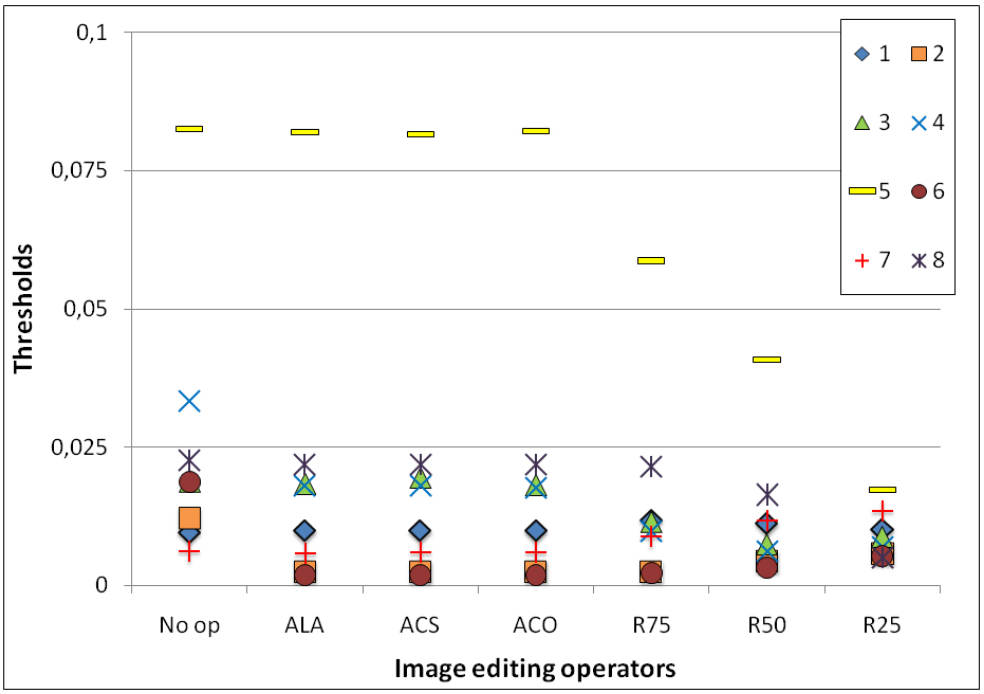

Figure 4.3: Thresholds values used according to the photo editing operations being tested.

4.4.3 Experiment 3 - Online photo sharing

In this experiment, the Lukáš *et al.* identification technique has been experimented by applying it on pictures previously uploaded on each of OSNs websites. As described

in section 4.3.3 the experiment was conducted in two phases. In the first one, we have used the thresholds computed in the experiment section 4.4.1 and in the second one the thresholds are recalculated on the downloaded images.

Confidence measure of experiment 1

The results, presented in Table 4.7, show a substantial failure of the identification technique. On a side, this was expectable because, as already noted in the previous section, thresholds evaluated using unmodified images imply bad performance when classifying modified pictures.

On the other side, however, it is worth to note that these bad performances are also much worser than the one experienced on resized pictures in experiment 1 (see 4.4.1) and experiment 2 (see 4.4.2), especially when processing pictures retrieved from FAC and MSP. This seems to suggest that the reason of this behavior is not only the resizing of the processed images but also to other factors such as, for example, compression tricks of the images retrieved from the considered OSNs/OPSs.

	OSN / OPS service			
ID	**No**	**FAC**	**PHB**	**MSP**
1	—	178	96	180
2	—	176	80	180
3	9	171	40	180
4	—	159	46	180
5	—	180	104	180
6	—	180	178	180
7	2	2	2	20
8	—	178	22	180
Tot.	11	1224	568	1280

Table 4.7: Number of images rejected on pictures previously uploaded on a OSN / OPS with thresholds computed as in Experiment 1 (Resize, 4.4.1) .

Confidence measure recalculated

The results, shown in Table 4.8, confirm a strong improvement of the identification technique with respect to the previous test when analyzing pictures retrieved from FAC and PHB. This is especially the case of pictures taken using camera with ID 3 and 4. A fair improvement is also evident for pictures taken using cameras with ID 5, 6 and 8. Instead, the identification is mostly uneffective when processing pictures retrieved from MSP. These differences are probably due to the different compression strategies employed by the considered OSNs/OPSs when uploading pictures. MSP is likely to be the service that adopts the most aggressive strategy, as witnessed by results presented in Table 4.2.

	FAC		**PHB**		**MSP**	
ID	Decision threshold	Images rejected (FRR)	Decision threshold	Images rejected (FRR)	Decision threshold	Images rejected (FRR)
1	0,0062	165 (0,9133)	0,0057	36 (0,2000)	0,0118	180 (1)
2	0,0076	175 (0,9667)	0,0090	105 (0,5833)	0,0069	179 (0,9944)
3	0,0063	18 (0,0933)	0,0067	11 (0,0611)	0,0073	152 (0,8444)
4	0,0062	2 (0,0133)	0,0073	2 (0,0111)	0,0098	168 (0,9333)
5	0,0061	14 (0,0600)	0,0193	—	0,0062	160 (0,8888)
6	0,0040	84 (0,4867)	0,0035	10 (0,0556)	0,0049	179 (0,9944)
7	0,0072	4 (0,0200)	0,0070	2 (0,0111)	0,0078	90 (0,5000)
8	0,0069	68 (0,3867)	0,0080	2 (0,0111)	0,0112	180 (1)
Tot.		530		168		1288

Table 4.8: Decision thresholds, FRR and number of images rejected on the red channel for the tests FAC, PHB and MSP.

4.5 Experimental analysis of the SVM identification technique

In this section we are going to present the results of experiments described in 4.3 by using the SVM identification technique.

4.5.1 Experiment 1 - How to correlate images of different size?

In the Table 4.9 we present the confusion matrix of the experiment SVM-SUB. The confusion matrix is a matrix used to represent errors in assigning classes to observed patterns in which the element in the row i and column j represents the number of samples from class i which were classified as class j. Since there are a lot of confusion matrices, for readability are published in Appendix A and in this section we presents the results for each experiment in terms of images wrongly identified by the classifier.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	144	—	—	—	—	—	**6**	—
	2	—	147	**1**	—	—	—	**2**	—
	3	—	—	150	—	—	—	—	—
	4	—	—	**7**	143	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	**4**	**5**	—	—	140	**1**	—
	7	—	—	—	—	—	—	150	—
	8	—	**1**	**3**	—	—	—	—	146
Accuracy: 97,5% (1170 on 1200)									

Table 4.9: Confusion matrix for experiment SVM-Sub.

As shown in Table 4.10 also with this identification technique the best approach to make two images of the same size is the **Resize** while the worst is **Sub**. The reasons for this performance are the same as presented in the section 4.4.1, namely the elimination of a large part of the original image, when processing large pictures.

In all the remaining experiments presented in this thesis with the SVM identification technique, when needed, the **Resize** technique will be used.

ID	**SVM-Sub**	**SVM-Crop**	**SVM-Resize**
1	6	4	—
2	3	—	—
3	—	1	—
4	7	—	—
5	—	—	—
6	10	6	6
7	—	—	—
8	4	4	1
Tot.	30	15	7

Table 4.10: Number of images classified erroneously for experiments SVM-Sub, SVM-Crop and SVM-Resize.

4.5.2 Experiment 2 - Photo editing

As described in section 4.3.2 the experiment was conducted in two phases. In the first one, we have used the training model computed in the experiment section 4.5.1 and in the second one the training model recalculated on the processed images.

Confidence measure of experiment 1

The results of this experiment, presented in Table 4.11, are notable. A small increase in the number of erroneously identified images can be observed when considering pictures processed with the ALA, ACS and ACO operations. This increase is much more significant when considering the resized images as shown in Table 4.12. In this second group of operations, the number of wronged identified pictures is high and grows linearly with the resize factor.

By examining in details these results, it is worth noting that there are some camera models where the identification method performs very poorly when used with the re-

ID	**SVM-ALA**	**SVM-ACS**	**SVM-ACO**
1	1	—	—
2	2	2	2
3	—	—	—
4	2	2	2
5	—	—	—
6	13	13	14
7	—	—	—
8	4	4	4
Tot.	22	21	22

Table 4.11: Number of images classified erroneously for experiments SVM-ALA, SVM-ACS and SVM-ACO with training set of experiment SVM-Resize.

ID	**SVM-R75**	**SVM-R50**	**SVM-R25**
1	—	—	5
2	2	2	3
3	3	7	7
4	2	2	18
5	—	1	135
6	16	28	148
7	—	—	—
8	1	6	69
Tot.	24	46	385

Table 4.12: Number of images classified erroneously for experiments SVM-R75, SVM-R50 and SVM-R25 with training set of experiment SVM-Resize.

sized images such as the models 5, 6 and 8. It can be noted that if the training model is based on pictures that have not been previously processed, the identification technique may likely fail.

Confidence measure recalculated

In the previous test, it can be observed that when trying to classify processed pictures using a classifier that has been tuned for unmodified pictures, the identification technique may fail. The failures are mostly due to the alteration of the pattern noise existing in a processed image. This alteration implies a wrong classification. A natural solution to this problem consists in re-train the classifier using as training set the processed images.

The results, presented in Table 4.13, are similar to the results shown in the previous test. This results shows how the ALA, ACS and ACO operations did not influence the classification by the SVM.

ID	**SVM-ALA**	**SVM-ACS**	**SVM-ACO**
1	1	—	—
2	2	2	2
3	—	—	—
4	2	2	2
5	—	—	—
6	13	13	13
7	—	—	—
8	4	4	4
Tot.	22	21	21

Table 4.13: Number of images classified erroneously for experiments SVM-ALA, SVM-ACS and SVM-ACO with training set recalculated

The results of the identification technique, documented in Table 4.14 with a re-trained classifier on resized images show a significant improvement in the quality of the classification, with respect to the previous test over these images. Note that in this test the R25 operation shows the best improvement with respect to previous test.

ID	**SVM-R75**	**SVM-R50**	**SVM-R25**
1	—	—	—
2	2	2	2
3	—	—	7
4	2	2	—
5	—	—	—
6	12	12	15
7	1	1	3
8	1	3	11
Tot.	18	20	38

Table 4.14: Number of images classified erroneously for experiments SVM-R75, SVM-R50 and SVM-R25 with training set recalculated

4.5.3 Experiment 3 - Online photo sharing

As described in section 4.3.3 the experiment was conducted in two phases. In the first one, we have used the training model computed in the experiment section 4.5.1 and in the second one the training model recalculated on the processed images.

Confidence measure of experiment 1

The results, presented in Table 4.15, show a considerable failure of the identification technique, especially when processing pictures retrieved from FAC and MSP.

Confidence measure recalculated

The results, documented in Table 4.16, confirm a strong improvement of the identification technique with respect to the previous test when analyzing pictures retrieved from FAC and PHB. This is especially the case of pictures taken using camera with ID 5, 6 and 8. Instead, the identification is mostly uneffective when processing pictures retrieved form MSP.

ID	**SVM-FAC**	**SVM-PHB**	**SVM-MSP**
1	150	107	150
2	92	7	135
3	25	7	130
4	114	3	150
5	150	140	150
6	150	150	150
7	—	—	—
8	150	89	150
Tot.	831	503	1015

Table 4.15: Number of images classified erroneously for experiments SVM-FAC, SVM-PHB and SVM-MSP with training set of experiment SVM-Resize.

ID	**SVM-FAC**	**SVM-PHB**	**SVM-MSP**
1	7	—	58
2	15	—	71
3	6	3	26
4	—	2	31
5	1	—	31
6	10	9	29
7	3	1	27
8	4	1	56
Tot.	46	16	329

Table 4.16: Number of images classified erroneously for experiments SVM-FAC, SVM-PHB and SVM-MSP with training set recalculated

4.6 Comparison of the identification techniques

In this section we compare the performance of the SVM classifier against the method of Lukáš . As the testing of the SVM classifier is executed on 150 images, to properly compare the results we exclude from the image dataset of the Lukas method the 30 images, for camera, used for create the training sets of the SVM classifier, following we recalculate the thresholds and the number of images rejected.

4.6.1 Experiment 1 - How to correlate images of different size?

In the Table 4.17 are compared the performance of the identification techniques for the experiment 1. In the Sub and Crop tests SVM performs better particularly for camera with ID 7. In the Resize test, results of the SVM technique are inline with the results obtained by using method of Lukáš Ṅote that all, even if the total is similar, radically changes the camera that gets worse with both a technique and with the other.

	Sub		**Crop**		**Resize**	
ID	**Lukáš**	**SVM**	**Lukáš**	**SVM**	**Lukáš**	**SVM**
1	14	6	—	4	—	—
2	20	3	—	—	—	—
3	13	—	6	1	7	—
4	3	7	—	—	—	—
5	—	—	1	—	—	—
6	5	10	—	6	—	6
7	144	—	126	—	1	—
8	4	4	2	4	—	1
Tot.	203	30	135	15	8	7

Table 4.17: Number of images classified erroneously with the method of Lukáš *et al.* and the SVM classifier for experiment 1 (see 4.3.1).

4.6.2 Experiment 2 - Photo editing

In the Table 4.18 and Table 4.19 are compared the performance of the identification techniques for ALA, ACS and ACO operations of the experiment 2. In the first one the identification is performed using the thresholds and the training model computed in the experiment 1 with Resize. In the second one the thresholds and the training model is recalculated by using the modified images.

	ALA		**ACS**		**ACO**	
ID	**Lukáš**	**SVM**	**Lukáš**	**SVM**	**Lukáš**	**SVM**
1	—	1	—	—	—	—
2	3	2	3	2	3	2
3	8	—	11	—	10	—
4	2	2	2	2	3	2
5	—	—	—	—	1	—
6	8	13	5	13	7	14
7	1	—	2	—	1	—
8	—	4	1	4	1	4
Tot.	22	22	24	21	26	22

Table 4.18: Number of images classified erroneously with the method of Lukáš *et al.* and the SVM classifier for experiments 2 (see 4.3.2, ALA, ACS and ACO) with thresholds and training model computed in experiment 1 (see 4.3.1, Resize)

.

In both cases, the total number of images correctly identified by both techniques are in line and recalculation of the confidence measure is useless.

The performance of the identification techniques for R75, R50 and R25 operations, shown in tables 4.20 and 4.21, seen the SVM in slightly difficulty to maintain the same number of recognitions but the wrong number of errors is very close. Both techniques show us that the number of wronged identified pictures grows linearly with the resize factor.

	ALA		ACS		ACO	
ID	**Lukáš**	**SVM**	**Lukáš**	**SVM**	**Lukáš**	**SVM**
1	—	1	—	—	—	—
2	2	2	2	2	2	2
3	9	—	9	—	9	—
4	1	2	1	2	1	2
5	—	—	—	—	—	—
6	5	13	5	13	5	13
7	1	—	1	—	—	—
8	—	4	—	4	—	4
Tot.	18	22	18	21	17	21

Table 4.19: Number of images classified erroneously with the method of Lukáš *et al.* and the SVM classifier for experiments 2 (see 4.3.2, ALA, ACS and ACO) with thresholds and training model recalculated on processed images

.

	R75		R50		R25	
ID	**Lukáš**	**SVM**	**Lukáš**	**SVM**	**Lukáš**	**SVM**
1	—	—	—	—	2	5
2	2	2	2	2	13	3
3	9	3	14	7	46	7
4	3	2	7	2	50	18
5	—	—	1	1	103	135
6	9	16	10	28	94	148
7	1	—	2	—	2	—
8	2	1	2	6	37	69
Tot.	26	24	38	46	347	385

Table 4.20: Number of images classified erroneously with the method of Lukáš *et al.* and the SVM classifier for experiments 2 (see 4.3.2, R75, R50 and R25) with thresholds and training model recalculated in experiment 1 (see 4.3.1, Resize).

	R75		**R50**		**R25**	
ID	**Lukáš**	**SVM**	**Lukáš**	**SVM**	**Lukáš**	**SVM**
1	—	—	—	—	2	—
2	2	2	—	2	7	2
3	9	—	—	—	8	7
4	1	2	1	2	2	—
5	—	—	—	—	—	—
6	5	12	6	12	11	15
7	1	1	1	1	1	3
8	—	1	—	3	4	11
Tot.	18	18	8	20	35	38

Table 4.21: Number of images classified erroneously with the method of Lukáš *et al.* and the SVM classifier for experiments 2 (see 4.3.2, R75, R50 and R25) with thresholds and training model recalculated on processed images

.

4.6.3 Experiment 3 - Online photo sharing

In the Tables 4.22 and 4.23 are compared the performance of the identification techniques for FAC, PHB and MSP online social networsk. In the first one the identification is performed using the thresholds and the training model computed in the experiment 1 with Resize. In the second one the thresholds and the training model is recalculated by using the downloaded images.

	FAC		**PHB**		**MSP**	
ID	**Lukáš**	**SVM**	**Lukáš**	**SVM**	**Lukáš**	**SVM**
1	149	150	78	107	150	150
2	146	92	65	7	150	135
3	143	25	32	7	150	130
4	130	114	31	3	150	150
5	150	150	78	140	150	150
6	150	150	148	150	150	150
7	1	—	1	—	17	—
8	148	150	19	89	150	150
Tot.	1017	831	452	503	1067	1015

Table 4.22: Number of images classified erroneously with the method of Lukáš *et al.* and the SVM classifier for experiments 3 (see 4.3.3, FAC, PHB and MSP) with thresholds and training model computed in experiment 1 (see 4.3.1, Resize)

Results shows that SVM technique performs better than Lukáš *et al.* on images strongly decreased in quality as those downloaded from FAC and MSP. This trend is more noticeable in the test with the thresholds and training model recalculated. Here the results are fine when compared to the SVM method Lukas.

The bad performance of the identification techniques on pictures retrieved from OSNs/OPSs may also be due to other reasons, apart from the image compression. As a matter of fact, it can not be excluded that OSNs/OPSs may add some kind of "watermarking" to all the photos that flow on their websites. Such possibility could have two opposite effects from the Image Forensics point of view. On the one hand,

	FAC		PHB		MSP	
ID	**Lukáš**	**SVM**	**Lukáš**	**SVM**	**Lukáš**	**SVM**
1	137	7	30	—	150	58
2	145	15	84	—	149	71
3	14	6	7	3	126	26
4	2	—	2	2	143	31
5	9	1	—	—	134	31
6	73	10	9	9	149	29
7	3	3	1	1	76	27
8	58	4	1	1	150	56
Tot.	441	46	134	16	1077	329

Table 4.23: Number of images classified erroneously with the method of Lukáš *et al.* and the SVM classifier for experiments 3 (see 4.3.3, FAC, PHB and MSP) with thresholds and training model recalculated on processed images

the inscription of a watermark on a picture could alter its inner structure and fool the identification process, thus leading to a wrong classification. On the other hand, if the pictures have been previously "watermarked" by the OSNs/OPSs, the eventual "discovery" of the adopted watermarking technique could give useful hints in the direction of establishing which is the OSN/OPS service that hosted the image under scrutiny. In this case, what will be assessed is the OSN/OPS that processed the image and not which camera model were adopted to shoot the photo.

Chapter 5

Conclusions

In this thesis we present our work about the problem of the *source camera identification*. This problem is especially relevant in the digital forensic science, where determining the source of a given digital image can be critical in many trials.

The source camera identification requires to understand the processes involved in the creation of the digital images as well as the identification of the factors which could support or prevent the source camera identification. In particular, a key factor supporting the camera identification is the image noise, a variation of the digital signal characteristic of each camera. On the other hand, a critical issue able to prevent the source camera identification is the tampering of the image, which can alter the data enough to make the identification process ineffective.

In this thesis is presented an extended experimental evaluation of one of the most effective source camera identification techniques proposed so far, by Lukáš *et al.*; moreover, here is presented a new technique for classification of images to the right source camera via an SVM classifier. Both methods uses the characteristic noise left by the sensor on a digital picture as a fingerprint in order to identify the source camera used to take the picture.

The aim of the experiments is to assess the effectiveness of this techniques when

used with images previously modified using several common image-processing functions coming with photo-editing tools. Moreover, the techniques are applied to photos passed through Online Social Networks (OSN) or Online Photo Sharing (OPS) websites, without any human modification but only elaborated by such Web 2.0 tools.

The results confirm that, in several cases, the methods are resilient to the modifications introduced by the considered image-processing functions. However, in the experiments it has been possible to identify several cases where the quality of the identification process was deteriorated because of the noise introduced by the image-processing. In addition, when dealing with Online Social Networks and Online Photo Sharing services, it has been noted that some of them process and modify the uploaded pictures. These modifications make ineffective, in many cases, the methods. However, the results of the tests show that the classification of the altered images may perform very poorly if the classifiers have been tuned using unmodified images. This problem can be fixed by tuning the classifiers according to a data set of altered images. Consequently, in the method of Lukáš *et al.* by lowering the decision thresholds used to establish whether a picture has been taken with a given camera while ,in the SVM method, by defining a new training model. In this new configuration, the methods confirms their effectiveness, even when processing altered images. However, there are processing operations, such as resizing and/or increasing the compression factor of a JPEG picture, which seems to have a negative effect on the results of the classification. As a side effect of the tests, it was noticed that the use of a *Resize* operation seems to be preferable to a *Crop* one when calculating the correlation between two images of different sizes.

The decrease of the threshold involves, nonetheless, several problems while choosing which one to use during a real investigation on a photographic exhibit. In fact, if the threshold computation is performed on a set of "unaltered" images, then the obtained threshold will be greater than the correlation index of a given, altered, image under

scrutiny. Otherwise, if the computation of the decision threshold is computed on a set of altered images then the FRR is increased.

The research also investigates if and how OSN/OPS services modify the images that transit on their websites, and tries to establish if the methods presented in this thesis are able to correctly identify images modified in this way. The investigation has been conducted by means of three tests. The first determines which OSN/OPS, among a set of them containing some of the most popular ones, alters the pictures.

In the second test, the source camera identification methods are applied on photos that have been previously uploaded on the OSNs/OPSs scrutinized in the above step. The results show a significant inadequacy of the identification methods when using the same decision threshold, and training model, computed for original images.

The last test performed on the OSN/OPS systems has been conducted on the same data set of the second test, but using threshold values and training model computed on images stored and downloaded from the examined OSNs/OPSs. The results validate and confirm the expectations that the identification technique behaves better, especially the SVM method, when using a threshold or training model that have been extracted in the "correct" way. Despite this, the identification revealed to be unsuccessful in many cases, mostly because of the compression strategies employed by the considered OSNs/OPSs in order to reduce the size of uploaded images.

Another factor, that can be further analyzed in a future work, is that the OSNs/OPSs could watermark in some way the images that flow on their websites. The watermarking operations while from one hand could contribute to the bad performance of the Lukáš *et al.* method, on the other hand could give useful hints in the direction of establishing the OSN/OPS that handled a given photo. In a few words, it would be possible to distinguish if a given photo under investigation has been posted to an OSN/OPS website just analyzing some "hidden" characteristics of the photo without relying on its "evident" origin.

References

[1] Adobe Photoshop Web Site. http://www.adobe.com/products/photoshop/, June 2010. 40

[2] Sevinc Bayram, Husrev T. Sencar, Nasir D. Memon, and Ismail Avcibas. **Source Camera Identification Based on CFA Interpolation**. In *ICIP (3)*, pages 69–72, 2005. 25

[3] H. Blitzer, K. Stein-Ferguson, and J. Huang. *Understanding Forensic Digital Imaging*. Academic Press, 2008.

[4] Cambridge In Colour - Understanding digital camera sensors. http://www.cambridgeincolour.com/tutorials/camera-sensors.htm, April 2011. 15

[5] Camera Imaging Products Association. http://cipa.jp/english/index.html, January 2011. 1

[6] Camera Labs. http://www.cameralabs.com/reviews/Canon_EOS_1000D_Rebel_XS/verdict.shtml, June 2010. 36

[7] Eoghan Casey. *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet with Cdrom*. Academic Press, Inc., Orlando, FL, USA, 1st edition, 2000. 7

[8] Aniello Castiglione, Giuseppe Cattaneo, Maurizio Cembalo, and Umberto Ferraro Petrillo. **Source Camera Identification in Real Practice: A Preliminary Experimentation**. In *BWCCA*, pages 417–422, 2010.

[9] Aniello Castiglione, Giuseppe Cattaneo, Maurizio Cembalo, and Umberto Ferraro Petrillo. **Experimentations with Source Camera Identification and Online Social Networks**. 2011.

[10] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 35

[11] Mo Chen, Jessica J. Fridrich, Miroslav Goljan, and Jan Lukás. **Determining Image Origin and Integrity Using Sensor Noise**. *IEEE Transactions on Information Forensics and Security*, **3**(1):74–90, 2008. 25

[12] K. S. Choi, E. Y. Lam, and K. K. Y. Wong. **Source camera identification using footprints from lens aberration**. In *Proceedings of the SPIE*, pages 69–72, 2006. 25

[13] Chris Putnam. `http://blog.facebook.com/blog.php?topic_id=185341929641`, June 2010.

[14] Frederick B. Cohen. **Fundamentals of Digital Forensic Evidence**. In *Handbook of Information and Communication Security*, pages 789–808. 2010.

[15] Daubert v. Merrell - Dow Pharmaceuticals Inc., . `http://www.law.cornell.edu/supct/html/92-102.ZO.html`, 1993. 7

[16] DFRWS. **A roadmap for digital forensics research**. Technical report, Digital Forensic Research Workshop, November 2001. Report From the First Digital Forensic Research Workshop (DFRWS). 5

[17] Miroslav Goljan. **Digital Camera Identification from Images - Estimating False Acceptance Probability**. In *IWDW*, pages 454–468, 2008.

[18] Miroslav Goljan, Jessica J. Fridrich, and Tomas Filler. **Large Scale Test of Sensor Fingerprint Camera Identification**. *Proc. SPIE, Electronic Imaging, Media Forensics and Security XI*, 2009. 25, 26

[19] Google Picasa. `http://picasatutorials.com/2009/04/picasa-tip-im-feeling-lucky/`, June 2010. 2

[20] Hitwise Intelligenze - MySpace moves into first position . `http://weblogs.hitwise.com/bill-tancer/2006/07/myspace_moves_into_1_position.html`, March 2011.

[21] Gerald C. Holst. *CCD arrays, cameras, and displays*. JCD Pub. ; SPIE Optical Engineering Press, 1996.

[22] C. Holt. **Two-channel likelihood detectors for arbitrary linear channel distortion**. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, **35**(3):267 – 273, 1987.

[23] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. **A Practical Guide to Support Vector Classification**, 2000.

[24] James R. Janesick and Morley Blouke. **Scientific Charge-Coupled Devices: Past, Present, & Future**. *Opt. Photon. News*, **6**(4):16–20, 1995.

[25] Mehdi Kharrazi, Husrev T. Sencar, and Nasir D. Memon. **Blind Source Camera Identification**. In *ICIP*, pages 709–712, 2004. 25

[26] M. Kivanc Mihcak, I. Kozintsev, and K. Ramchandran. **Spatially Adaptive Statistical Modeling of Wavelet Image Coefficients and its Application to Denoising**. In *ICASSP '99: Proceedings of the Acoustics, Speech, and*

Signal Processing, IEEE International Conference, pages 3253–3256, Washington, DC, USA, 1999. IEEE Computer Society. 27, 34

[27] Tran Van Lanh, Kai-Sen Chong, Sabu Emmanuel, and Mohan S. Kankanhalli. **A Survey on Digital Camera Image Forensic Methods**. In *ICME*, pages 16–19, 2007.

[28] Yangjing Long and Yizhen Huang. **Image Based Source Camera Identification using Demosaicking**. In *Multimedia Signal Processing, 2006 IEEE 8th Workshop on*, pages 419 –424, 2006. 26

[29] Jan Lukás, Jessica J. Fridrich, and Miroslav Goljan. **Digital Camera Identification from Sensor Pattern Noise**. *IEEE Transactions on Information Forensics and Security*, **1**(2):205–214, 2006. 25, 33

[30] Weiqi Luo, Zhenhua Qu, Feng Pan, and Jiwu Huang. **A of passive technology for digital image forensics**. *Frontiers of Computer Science in China*, **1**(2):166–179, 2007.

[31] MathWorks MATLAB Web Site. `http://www.mathworks.com/products/matlab/`, June 2010. 34

[32] M. Kivan Mihak, Igor Kozintsev, and Kannan Ramchandran. **Spatially Adaptive Statistical Modeling Of Wavelet Image Coefficients And Its Application To Denoising**. In *in Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc*, pages 3253–3256, 1999.

[33] Kara L. Nance and Daniel J. Ryan. **Legal Aspects of Digital Forensics: A Research Agenda**. In *HICSS*, pages 1–6, 2011. 7

[34] Photobucket About us page . `http://photobucket.com/about`, March 2011. 43

[35] Royal Pingdom. http://royal.pingdom.com/2010/01/22/internet-2009-in-numbers/, June 2010.

[36] H. T. Sencar and N. Memon. **Overview of State-of-the-Art in Digital Image Forensics**.

Appendix A

Confusion matrices of SVM experiments

In this appendix we present all the confusion matrices of the SVM experimental analysis described in section 4.5.

A confusion matrix is an n-dimensional square matrix, where n is the number of distinct target values. The row indexes of a confusion matrix correspond to actual values observed and used for model testing. The column indexes correspond to predicted values produced by applying the model to the test data. For any pair of actual/predicted indexes, the value indicates the number of records classified in that pairing.

A.1 Experiment 1 - How to correlate images of different size?

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	144	—	—	—	—	—	**6**	—
	2	—	147	**1**	—	—	—	**2**	—
	3	—	—	150	—	—	—	—	—
	4	—	—	**7**	143	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	**4**	**5**	—	—	140	**1**	—
	7	—	—	—	—	—	—	150	—
	8	—	**1**	**3**	—	—	—	—	146
Accuracy: 97,5% (1170 on 1200)									

Table A.1: Confusion matrix for experiment SVM-Sub.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	146	—	—	—	—	—	**4**	—
	2	—	150	—	—	—	—	—	—
	3	—	—	149	—	—	—	**1**	—
	4	—	—	—	150	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	**4**	—	—	—	144	**2**	—
	7	—	—	—	—	—	—	150	—
	8	—	**4**	—	—	—	—	—	146
Accuracy: 98,75% (1185 on 1200)									

Table A.2: Confusion matrix for experiment SVM-Crop.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	150	—	—	—	—	—	—
	3	—	—	150	—	—	—	—	—
	4	—	—	—	150	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	—	—	—	—	144	**6**	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	**1**	149
Accuracy: 99,4166% (1193 on 1200)									

Table A.3: Confusion matrix for experiment SVM-Resize.

A.2 Experiment 2 - Photo editing

Confidence measure of experiment 1

		Predicted							
	ID	1	2	3	4	5	6	7	8
Actual	1	149	—	—	—	—	—	**1**	—
	2	—	148	—	—	—	—	**2**	—
	3	—	—	150	—	—	—	—	—
	4	—	—	**2**	148	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	—	—	—	—	137	**13**	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	**4**	146
Accuracy: 98,1667% (1178 on 1200)									

Table A.4: Confusion matrix for experiment SVM-ALA with training set of experiment SVM-Resize.

		Predicted							
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	148	—	—	—	—	**2**	—
	3	—	—	150	—	—	—	—	—
	4	—	—	**2**	148	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	—	—	—	—	137	**13**	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	**4**	146
Accuracy: 98,25% (1179 on 1200)									

Table A.5: Confusion matrix for experiment SVM-ACS with training set of experiment SVM-Resize.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	148	—	—	—	—	**2**	—
	3	—	—	150	—	—	—	—	—
	4	—	—	**2**	148	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	—	—	—	—	136	**14**	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	**4**	146

Accuracy: 98,1667% (1178 on 1200)

Table A.6: Confusion matrix for experiment SVM-ACO with training set of experiment SVM-Resize.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	148	—	—	—	—	**2**	—
	3	—	—	147	—	—	—	**3**	—
	4	—	—	—	148	—	—	**2**	—
	5	—	—	—	—	150	—	—	—
	6	—	—	—	—	—	134	**16**	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	**1**	149

Accuracy: 98% (1176 on 1200)

Table A.7: Confusion matrix for experiment SVM-R75 with training set of experiment SVM-Resize.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	148	—	—	—	—	**2**	—
	3	—	—	143	—	—	—	**7**	—
	4	—	—	—	148	—	—	**2**	—
	5	—	—	—	—	149	—	**1**	—
	6	—	—	—	—	—	122	**28**	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	**6**	144

Accuracy: 96,1667% (1154 on 1200)

Table A.8: Confusion matrix for experiment SVM-R50 with training set of experiment SVM-Resize.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	145	—	—	—	—	—	**5**	—
	2	—	147	—	—	—	—	**3**	—
	3	—	—	143	—	—	—	**7**	—
	4	—	—	**10**	132	—	—	**8**	—
	5	—	**1**	—	—	15	—	**134**	—
	6	—	—	—	—	—	2	**148**	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	**69**	81
Accuracy: 67,9167% (815 on 1200)									

Table A.9: Confusion matrix for experiment SVM-R25 with training set of experiment SVM-Resize.

Confidence measure recalculated

		Predicted							
	ID	1	2	3	4	5	6	7	8
Actual	1	149	—	—	—	—	—	1	—
	2	—	148	—	—	—	—	2	—
	3	—	—	150	—	—	—	—	—
	4	—	—	2	148	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	—	—	—	—	137	13	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	4	146
Accuracy: 98.1667% (1178 on 1200)									

Table A.10: Confusion matrix for experiment SVM-ALA with training set recalculated.

		Predicted							
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	148	—	—	—	—	2	—
	3	—	—	150	—	—	—	—	—
	4	—	—	2	148	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	—	—	—	—	137	13	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	4	146
Accuracy: 98.25% (1179 on 1200)									

Table A.11: Confusion matrix for experiment SVM-ACS with training set recalculated.

		Predicted							
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	148	—	—	—	—	2	—
	3	—	—	150	—	—	—	—	—
	4	—	—	2	148	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	—	—	—	—	137	13	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	4	146
Accuracy: 98.25% (1179 on 1200)									

Table A.12: Confusion matrix for experiment SVM-ACO with training set recalculated.

		Predicted							
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	148	2	—	—	—	—	—
	3	—	—	150	—	—	—	—	—
	4	—	—	2	148	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	—	10	—	—	138	2	—
	7	—	—	1	—	—	—	149	—
	8	—	—	—	—	—	—	1	149
Accuracy: 98.5% (1182 on 1200)									

Table A.13: Confusion matrix for experiment SVM-R75 with training set recalculated.

		Predicted							
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	148	2	—	—	—	—	—
	3	—	—	150	—	—	—	—	—
	4	—	—	2	148	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	—	12	—	—	138	—	—
	7	—	—	1	—	—	—	149	—
	8	—	—	3	—	—	—	—	147
Accuracy: 98.3333% (1180 on 1200)									

Table A.14: Confusion matrix for experiment SVM-R50 with training set recalculated.

		Predicted							
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	148	—	2	—	—	—	—
	3	—	—	143	7	—	—	—	—
	4	—	—	—	150	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	—	1	5	9	—	135	—	—
	7	—	—	2	1	—	—	147	—
	8	—	1	3	7	—	—	—	139
Accuracy: 96.8333% (1162 on 1200)									

Table A.15: Confusion matrix for experiment SVM-R25 with training set recalculated.

A.3 Experiment 3 - Online photo sharing

Confidence measure of experiment 1

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	**0**	**2**	—	—	—	—	**148**	—
	2	—	58	—	—	—	—	**92**	—
	3	—	—	125	—	—	—	**25**	—
	4	—	**1**	**27**	36	—	—	**86**	—
	5	—	—	—	—	**0**	—	**150**	—
	6	—	—	—	—	—	**0**	**150**	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	**150**	**0**
Accuracy: 30,75% (369 on 1200)									

Table A.16: Confusion matrix for experiment SVM-FAC with training set of experiment SVM-Resize.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	43	—	—	—	—	—	**107**	—
	2	—	143	—	—	—	—	**7**	—
	3	—	—	143	—	—	—	**7**	—
	4	—	—	**1**	147	—	—	**2**	—
	5	—	**5**	**56**	—	10	—	**79**	—
	6	—	—	—	—	—	**0**	**150**	—
	7	—	—	—	—	—	—	150	—
	8	—	—	—	—	—	—	**89**	61
Accuracy: 58,0833% (697 on 1200)									

Table A.17: Confusion matrix for experiment SVM-PHB with training set of experiment SVM-Resize.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	**0**	—	—	—	—	—	**150**	—
	2	—	15	**1**	—	—	—	**134**	—
	3	—	**2**	20	—	—	—	**128**	—
	4	—	**2**	**1**	**0**	—	—	**147**	—
	5	—	—	—	—	**0**	—	**150**	—
	6	—	—	—	—	—	**0**	**150**	—
	7	—	—	—	—	—	—	150	—
	8	—	**1**	—	—	—	—	**149**	**0**
Accuracy: 15,4167% (185 on 1200)									

Table A.18: Confusion matrix for experiment SVM-MSP with training set of experiment SVM-Resize.

Confidence measure recalculated

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	143	5	—	—	1	—	—	1
	2	12	135	—	—	—	2	—	1
	3	5	1	144	—	—	—	—	—
	4	—	—	—	150	—	—	—	—
	5	—	1	—	—	149	—	—	—
	6	9	—	—	—	—	140	—	1
	7	3	—	—	—	—	—	147	—
	8	1	2	—	—	—	1	—	146
Accuracy: 96.1667% (1154 on 1200)									

Table A.19: Confusion matrix for experiment SVM-FAC with training set recalculated.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	150	—	—	—	—	—	—	—
	2	—	150	—	—	—	—	—	—
	3	1	2	147	—	—	—	—	—
	4	—	2	—	148	—	—	—	—
	5	—	—	—	—	150	—	—	—
	6	1	7	1	—	—	141	—	—
	7	—	1	—	—	—	—	149	—
	8	1	—	—	—	—	—	—	149
Accuracy: 98.6667% (1184 on 1200)									

Table A.20: Confusion matrix for experiment SVM-PHB with training set recalculated.

	Predicted								
	ID	1	2	3	4	5	6	7	8
Actual	1	92	33	1	1	8	10	1	4
	2	10	79	2	6	7	29	1	16
	3	8	5	124	—	—	7	—	6
	4	4	5	1	119	4	15	—	2
	5	9	1	—	—	119	18	—	3
	6	8	5	1	—	7	121	—	8
	7	5	7	—	—	5	4	123	6
	8	15	19	1	1	3	17	—	94
Accuracy: 72.5833% (871 on 1200)									

Table A.21: Confusion matrix for experiment SVM-MSP with training set recalculated.

Appendix B

MATLAB code

B.1 ComputePRNU.m

This file implements the extraction of the PRNU from an image.

```
1  function [filteredImg]=ComputePRNU(img,sigma0)
2  size_img=size(img);
3  [row,col]=size(size_img);
4  if(col == 2)
5      nChannels=1;
6  else
7      nChannels=3;
8  end
9  % Start
10 for num_Channel=1:nChannels
11     imgNoiseChannel=img(:,:,num_Channel);
12     [C,S]=wavedec2(imgNoiseChannel,4,'db8');
13         fprintf('Compute PRNU (db8): waiting ...
14 ..........Start function on channel %d\n',num_Channel);
15         for numLevel=1:4
16                 [cH,cV,cD]=detcoef2('all',C,S,numLevel);
17         cH2=cH.^2;
18         cV2=cV.^2;
```

```
19          cD2=cD.^2;
20          for numKernel=1:4
21              kernel=fspecial('average',
22                              [numKernel+numKernel+1 numKernel+numKernel+1]);
23              cHsigma2(:,:,numKernel)=imfilter(cH2,kernel,'replicate');
24              cVsigma2(:,:,numKernel)=imfilter(cV2,kernel,'replicate');
25              cDsigma2(:,:,numKernel)=imfilter(cD2,kernel,'replicate');
26              cHsigma2(:,:,numKernel)=cHsigma2(:,:,numKernel)-(sigma0^2);
27              cVsigma2(:,:,numKernel)=cVsigma2(:,:,numKernel)-(sigma0^2);
28              cDsigma2(:,:,numKernel)=cDsigma2(:,:,numKernel)-(sigma0^2);
29              [r,c,v]=find(cHsigma2(:,:,numKernel) < 0);
30              for m=1:length(r)
31                  cHsigma2(r(m),c(m),numKernel)=0;
32              end
33              [r,c,v]=find(cVsigma2(:,:,numKernel) < 0);
34              for m=1:length(r)
35                  cVsigma2(r(m),c(m),numKernel)=0;
36              end
37              [r,c,v]=find(cDsigma2(:,:,numKernel) < 0);
38              for m=1:length(r)
39                  cDsigma2(r(m),c(m),numKernel)=0;
40              end
41          end
42          MinVarianzaCH= MinimumVariance( cHsigma2(:,:,1),
43                                          cHsigma2(:,:,2),
44                                          cHsigma2(:,:,3),
45                                          cHsigma2(:,:,4));
46          MinVarianzaCV= MinimumVariance( cVsigma2(:,:,1),
47                                          cVsigma2(:,:,2),
48                                          cVsigma2(:,:,3),
49                                          cVsigma2(:,:,4));
50          MinVarianzaCD= MinimumVariance( cDsigma2(:,:,1),
51                                          cDsigma2(:,:,2),
```

```
52                                                  cDsigma2(:,:,3),
53                                                  cDsigma2(:,:,4));
54          cHden=cH.*(MinVarianzaCH./(MinVarianzaCH+(sigma0^2)));
55          cVden=cV.*(MinVarianzaCV./(MinVarianzaCV+(sigma0^2)));
56          cDden=cD.*(MinVarianzaCD./(MinVarianzaCD+(sigma0^2)));
57
58          C=getModifyC(C,cHden,cVden,cDden,S,numLevel);
59          clear cHsigma2 cVsigma2 cDsigma2 kernel MinVarianzaCH  MinVarianzaCV
60                  MinVarianzaCD cHden cVden cDden cH2 cV2 cD2 cH cV cD r c v;
61      end
62      filteredImg(:,:,num_Channel)=waverec2(C,S,'db8');
63  end
64
65  function [minvar]=MinimumVariance(A,B,C,D)
66          minvar =min(A,B);
67          minvar =min(minvar,C);
68          minvar =min(minvar,D);
69  end
```

B.2 getResidulaNoise.m

This function returns, give an image, the image filtered with the PRNU filter and the residual noise.

```
1  function [residual_noise,filter_image]= getResidualNoise(Image)
2  % Gives an image, this function returns the image filtered  with the PRNU
3  % filter and the residual noise.
4
5  filter_image=ComputePRNU(Image,5);
6  residual_noise=double(Image)-double(filter_image);
```

B.3 ComputeReferencePattern.m

This function returns an image representing the residual noise that is the average of several residual noise obtained with the function `getResidualNoise`.

```
1  function [referencePattern]=ComputeReferencePattern(path, numbersOfImages)
2
3  % Return the average of the  residual noise
4  % on #numbersOfImages images in the directory path
5
6  flag=0;
7  numeroImmagini=0;
8  directory = dir(path);
9  files = {directory.name};
10
11 for i = 1 : length(files)
12
13     if i > (numbersOfImages+2)
14         break;
15     end
16
17     if ~directory(i).isdir
18         numeroImmagini=numeroImmagini+1;
19         if (flag==0)
20             fprintf('Compute Photo %d\n',i-2);
21             img=imread(strcat(percorso,char(files(i))));
22             [residual_noise]=getResidualNoise(img);
23             immagineMedia=residual_noise;
24             flag=1;
25             clear img residual_noise;
26         else
27             fprintf('Compute Photo %d\n',i-2);
28             img=double(imread(strcat(percorso,char(files(i)))));
29             [residual_noise]=getResidualNoise(img);
```

```
30                immagineMedia=immagineMedia+residual_noise;
31                clear img residual_noise;
32            end
33        end
34    end
35    immagineMedia=immagineMedia./numeroImmagini;
36    referencePattern=immagineMedia;
```

B.4 correlation.m

This file computes the correlation pixel-by-pixel between two images of the same dimension.

```
1  function [corr] = correlation(img, patt)
2  % Computes the correlation between two images (matrix)
3
4  img=double(img);
5  patt=double(patt);
6
7  % check same dimensions
8  if size(img) ~= size(patt)
9      disp('dimesions not equal');
10     return
11 end
12
13 [r1,c1,nc1]=size(img);
14 [r2,c2,nc2]=size(patt);
15
16 if(r1~=r2 || c1~=c2 || nc1~=nc2)
17     disp('dimesions not equal');
18 end;
19
20 size_img = size(img);
```

```
21  [row,col] = size(size_img);
22  if(col == 2)
23      nCanali = 1;
24  else
25      nCanali = 3;
26  end
27  corr = zeros(1,nCanali);
28  for canale = 1 : nCanali
29      media_img = mean(mean(img(:,:,canale)));
30      media_patt = mean(mean(patt(:,:,canale)));
31
32      somma = 0;
33      for i = 1 : size(img,1)
34          for j = 1 : size(img,2)
35              somma = somma + (img(i,j,canale) - media_img) *
36                              (patt(i,j,canale) - media_patt);
37          end
38      end
39
40      somma1 = 0;
41      somma2 = 0;
42      for i = 1 : size(img, 1)
43          for j = 1 : size(img, 2)
44              somma1 = somma1 + (img(i,j,canale) - media_img)^2;
45              somma2 = somma2 + (patt(i,j,canale) - media_patt)^2;
46          end
47      end
48      den = sqrt(somma1) * sqrt(somma2);
49      corr(1,canale) = somma / den;
50  end
```

B.5 ComputeCorrelations.m

This script computes the correlation between all the images found in a given directory folder and its subdirectory and the reference pattern of the cameras previously computed. It produces a comma separated value formatted file.

```
1  function computeCorrelations()
2  % input folder
3  folder='/img/ALA/'
4  NUMFILES=180;
5  TESTNAME='Correlations ALA';
6  BRIEFNAME='CORRESALA';
7
8  % reference pattern folder
9  RPFolder = '/img/RP/';
10 RPname = {   'RP_01_Canon EOS 400D.mat';
11               'RP_02_Canon EOS 1000D.mat';
12               'RP_03_Canon Powershot A400 istance A.mat';
13               'RP_04_Canon Powershot A400 istance B.mat';
14               'RP_05_Panasonic Lumix DMC-FZ20.mat';
15               'RP_06_Panasonic Lumix DMC-FS5.mat';
16               'RP_07_Kodak EasyShare Cx7530.mat';
17               'RP_08_HP Photosmart E327.mat'
18 };
19 RPStr = {    'RP(01-CanonEOS400D)';
20               'RP(02-CanonEOS1000D)';
21               'RP(03-CanonPS_A400_ist_A)';
22               'RP(04-CanonPS_A400_ist_B)';
23               'RP(05-PanasonicDMC-FZ20)';
24               'RP(06-PanasonicDMC-FS5)';
25               'RP(07-KodaxCX7530)';
26               'RP(08-HPE327)'
27 };
28 RPID = {     '1';'2';
```

```
29                  '3';'4';
30                  '5';'6';
31                  '7';'8'
32  };
33
34  DirIMAGES=folder;
35
36  fileLOG=strcat(folder,'logALA.txt');
37  FileLOG=fopen(fileLOG,'a+');
38  FileTIMINGCorr=fopen(strcat(folder,'CORRALA_timing.txt'),'a+');
39  fprintf(FileTIMINGCorr,'RP; RN; TIMECorr;\n');
40
41  FileTIMINGLoad=fopen(strcat(folder,'LOADALA_timing.txt'),'a+');
42  fprintf(FileTIMINGLoad,'FILE; TIMELoad;\n');
43
44  fprintf(FileLOG, '%s: \n\nStart %s (%s)\n',datestr(now),TESTNAME, BRIEFNAME);
45  righeRP = zeros (size(RPID,1),1);
46  colonneRP = zeros (size(RPID,1),1);
47  % load reference patterns
48  for i=1:size(RPID,1)
49      id = char(RPID(i));
50      rpStr=char(RPStr(i));
51      t0=cputime;
52      s=sprintf('load %s\n', rpStr);
53      logEntry(FileLOG, s);
54      fileRP = strcat(RPFolder, char(RPname(i)));
55      load(fileRP, 'residualNoise');
56      eval(sprintf('rpID_%s = residualNoise;', id));
57      [righeRP(i), colonneRP(i), canali] = eval(sprintf('size(rpID_%s)',id));
58      clear residualNoise   canali
59      t1=cputime-t0;
60      s =  sprintf('%s; %d;\n', char(RPStr(i)), t1);
61      fprintf(FileTIMINGLoad,s);
```

```
62 end
63 clear RPFolder folder ans fileRP i id rpStr s t0 t1
64
65 logEntry(FileLOG,'Start iteration over images'' folder');
66 directory = dir(DirIMAGES);
67 files = {directory.name};
68 for i = 1 : length(files)
69     if directory(i).isdir
70         % skip . and ..
71         if (strcmp(char(files(i)), '.')==1)
72             continue;
73         end
74         if (strcmp(char(files(i)), '..')==1)
75             continue;
76         end
77         if (strcmp(char(files(i)), '.DS_Store')==1)
78             continue;
79         end
80         if (strcmp(char(files(i)), 'Thumbs.db')==1)
81             continue;
82         end
83         model=char(files(i));
84         s=strcat('Start correlation for camera ', model);
85         logEntry(FileLOG,s);
86
87         dirS=strcat(DirIMAGES,'/',model,'/');
88         for z=1:size(RPID,1)
89             id = char(RPID(z));
90             rpStr=char(RPStr(z));
91             s=sprintf('Creating correlation file %s',rpStr);
92             logEntry(FileLOG,s);
93             str=strcat('CorALA_', rpStr, '_MOD_',model);
94             eval(sprintf('FileCor_ID_%s = fopen(''%s/%s.txt'',''a+'');',
```

```
95                          id,dirS,str));
96              eval(sprintf('entryFile(FileCor_ID_%s)',id));
97          end
98          clear rpStr s str z id
99
100         logEntry(FileLOG,'Start iteration over images');
101         checkDir=dir(dirS);
102         filesDir = {checkDir.name};
103         cnt=0;
104
105         for j = 1 : length(filesDir)
106             filename=char(filesDir(j));
107             dot = regexp(filename,'\.');
108             switch(filename(dot+1:end))
109                 case {'jpg','jpeg', 'JPG', 'JPEG'}
110                     ; % ok
111                 otherwise
112                     continue;
113             end
114
115             cnt = cnt+1;
116             fprintf('Compute Photo %d <<%s>>  \n',cnt ,filename);
117             s=sprintf('Compute Photo: %s', filename);
118             logEntry(FileLOG,s);
119             img=imread(strcat(dirS,filename));
120             clear img;
121
122            % check if the correlation has been previously made
123             loadRN = false;
124             for z=1:size(RPID,1)
125                 id = char(RPID(z));
126                 fileTest = sprintf('%s%s.RP_%s.CORALAOK',dirS,filename,id);
127                 if ~(exist(fileTest,'file')==2)
```

```
128                     loadRN = true;
129                     break;
130                 end
131             end
132
133             if (loadRN)
134                 tt0=cputime;
135                 [residual_noise_img]=getLocalSavedResidualNoise(dirS, filename);
136                 tt1=cputime-tt0;
137                 fprintf(FileTIMINGLoad,'%s; %d;\n', filename, tt1);
138                 for z=1:size(RPID,1)
139                     id = char(RPID(z));
140                     fileOk = sprintf('%s%s.RP_%s.CORALAOK',dirS,filename,id);
141                     if ~(exist(fileOk,'file')==2)
142                         t0=cputime;
143                         eval(sprintf('[ corrR corrG corrB] = '
144                                      'computeCorrelations('
145                                      'righe, colonne, righeRP(z),'
146                                      'colonneRP(z),rpID_%s,residual_noise_img,'
147                                      'strcat(dirS,filename));',id));
148
149                         strData = sprintf('%s; RN(%s); %s; %s; %s',char(RPStr(z)),
150                                           filename, corrR, corrG, corrB);
151                         s1 = sprintf('fprintf(FileCor_ID_%s,',id);
152                         eval(sprintf('%s''%s\\n'');',s1, strData));
153
154                         F = fopen(fileOk,'wt');
155                         fclose(F);
156                         clear F fileOk;
157                         t1=cputime-t0;
158                         fprintf(FileTIMINGCorr,'%s; RN(%s); %d;\n',
159                                 char(RPStr(z)), filename, t1);
160                         s=sprintf('Correlation %s - Photo (%s) computed. Time: %d',
```

```
161                                         char(RPStr(z)), filename, t1);
162                             logEntry(FileLOG,s);
163                         else
164                                 s=sprintf('Correlation %s - has been previously made.',
165                                         char(RPStr(z)), filename);
166                                 logEntry(FileLOG,s);
167                         end
168                     end
169                 else
170                      s=sprintf('All correlation - Photo (%s) has been previously made.',
171                                 filename);
172                      logEntry(FileLOG,s);
173                 end
174                 clear residual_noise_img;
175                 if (cnt==NUMFILES)
176                     break;
177                 end
178             end
179         end
180     end
181     for z=1:size(RPID,1)
182                 id = char(RPID(z));
183                 eval(sprintf('fclose(FileCor_ID_%s);',id));
184     end
185     logEntry(FileLOG,'end correlations');
186 end
187
188
189 function [corrR corrG corrB]=computeCorrelations(righe, colonne, righeRP, colonneRP,
    rpInput, rnImg, pathImg)
190
191     if ~((righe==righeRP)&&(colonne==colonneRP))
192         minRighe=min(righe, righeRP);
```

```
193             minColonne=min(colonne, colonneRP);
194             rp = imresize(rpInput, [minRighe minColonne]); %resize
195             rnImg = imresize(rnImg, [minRighe minColonne]); %resize
196         end
197         result_corr =correlation(rp, rnImg);
198         clear  rp rnImg;
199         [corrR] = replaceSTR(sprintf('%g',result_corr(1)));
200         [corrG] = replaceSTR(sprintf('%g',result_corr(2)));
201         [corrB] = replaceSTR(sprintf('%g',result_corr(3)));
202 end
203
204 function logEntry(file, str)
205               fprintf(file, '%s: %s\n',datestr(now),str);
206           fprintf('%s\n',str);
207 end
208
209
210 function [residual_noise,filter_image]= getLocalSavedResidualNoise(percorso, nomeFile)
211     % Load a residual noise if available
212     nomeFileRN=strcat('RN_',nomeFile);
213     nomeFileRN=strcat(nomeFileRN,'.mat');
214
215     % Cerca all'interno della cartella dove     presente il file JPG.
216     RN_FILE=strcat(percorso, nomeFileRN);
217     if (exist(RN_FILE,'file')==2)
218         load(RN_FILE);
219         %residual_noise=0;
220         return
221     end
222
223     % Not available. Compute and save it
224     img=imread(strcat(percorso, nomeFile));
225     filter_image=filtro_PNU(img,5);
```

```
226         residual_noise=double(img)-double(filter_image);
227         filter_image = 0;
228         save(RN_FILE, 'residual_noise','filter_image');
229     end
```

Declaration

I herewith declare that I have produced this work without the prohibited assistance of third parties and without making use of aids other than those specified. Notions taken over directly or indirectly from other sources have been identified as such. Some results in this Thesis are present in the following papers.

- M. Cembalo, A. Castiglione, G. Cattaneo, U. Ferraro Petrillo, Experimentations with Source Camera Identification and Online Social Networks, (to appear).
- M. Cembalo, A. Castiglione, G. Cattaneo, U. Ferraro Petrillo, Source Camera Identification in Real Practice: A Preliminary Experimentation, In BWCCA, pages 417-422, 2010.

Fisciano (SA), Italy