12-1-2004

Statistical Tools for Digital Image Forensics

Alin C. Popescu
Dartmouth College

Statistical Tools for Digital Image Forensics

A thesis submitted to the faculty
in partial fulfillment of the requirements for the
degree of Doctor of Philosophy in Computer Science

by

Alin C. Popescu

DARTMOUTH COLLEGE
Hanover, New Hampshire
December, 2004

Examining committee:	Hany Farid (chair)
	Bruce Randall Donald
	Daniel Rockmore
	Jana Kosecka
Dean of Graduate Students:	Charles K. Barlowe, Ph.D.

Dartmouth College
Department of Computer Science
Technical Report TR2005-531

This thesis is dedicated to my wife, Rosa Orellana, and to my parents, Floarea and Mihail-Viorel Popescu.

Abstract

Statistical Tools for Digital Image Forensics

Alin C. Popescu

A digitally altered image, often leaving no visual clues of having been tampered with, can be indistinguishable from an authentic image. The tampering, however, may disturb some underlying statistical properties of the image. Under this assumption, we propose five techniques that quantify and detect statistical perturbations found in different forms of tampered images: (1) re-sampled images (e.g., scaled or rotated); (2) manipulated color filter array interpolated images; (3) double JPEG compressed images; (4) images with duplicated regions; and (5) images with inconsistent noise patterns. These techniques work in the absence of any embedded watermarks or signatures. For each technique we develop the theoretical foundation, show its effectiveness on credible forgeries, and analyze its sensitivity and robustness to simple counter-attacks.

Acknowledgments

First and foremost I would like to thank my advisor, Hany Farid, for his advice and guidance over the years, and for helping me understand the importance of hard work and dedication in doing a Ph.D. thesis.

I would also like to thank all my professors at Dartmouth College, and my wonderful colleagues and friends. In particular, I would like to thank:

My thesis committee members: Bruce Randall Donald, Jana Kosecka, and Dan Rockmore,

My colleagues from the Image Science Group: Kimo Micah Johnson, Siwei Lyu, and Senthil Periaswamy,

My professors at Dartmouth: Jay Aslam, Tom Cormen, Doug McIlroy, Bill McKeeman, Carl Pomerance, Daniela Rus, and Cliff Stein,

The Sudikoff Lab system administrators: Sandy Brash, Wayne Cripps, John Konkle, and Tim Trebugov,

My friends and colleagues: Florin Constantin, Steve Linder, Virgil Pavlu, Luis Felipe Perrone, Geeta and Srdjan Petrovic, Tim Trebugov, and Anthony Yan.

Contents

Chapter 1

Introduction

During the past decade, powerful computers, high-resolution digital cameras, and sophisticated photo-editing software packages have become affordable and available to a large number of people. As a result, it has become fairly straightforward to create digital forgeries that are hard to distinguish from authentic photographs. These forgeries, if used in the mass media or courts of law, can have an important impact on our society. For example, a photograph taken during the 2003 Iraq war was published on the front page of the *Los Angeles Times*. This photograph, however, was not authentic: it was created by digitally splicing together two different photographs. The tampering was discovered by an editor at *The Hartford Courant* who noticed that some background people appeared twice in the photograph. Although the manipulation seems to have been merely intended to improve the composition of the photograph, the photojournalist responsible for it was fired. Another high profile case of a forged digital image that circulated on the internet in early 2004 was an image depicting Senator John Kerry and actress Jane Fonda sharing a stage at a peace rally against the Vietnam war[1]. The image made quite an impact, as Senator John Kerry was running for President of the United States and his involvement in the anti-war movement came under attack. This photograph was also created by digitally splicing together two separate images, and was exposed as a forgery when the photographer that took one of the original images came forward. These incidents, and many others, lead us to question the authenticity of the plethora of digital images that we are exposed to every day.

1.1 Image Tampering

Tampering with images is something neither new, nor recent. Some of the most well known examples of falsification of film photographs, for example, date back to the early years of the former Soviet Union, where both Lenin and Stalin had "the enemies of the people" removed from historic records, Figure 1.1. These, and similar, forgeries were created using image manipulations such as airbrushing, re-touching, dodging and burning, and contrast and color adjustment. Airbrushing

[1] Jane Fonda was a well known and controversial figure with strong views against the involvement of the United States in the war. After serving in the war, John Kerry spoke against it and got involved in the anti-war movement

Figure 1.1: Shown on the left are original photographs of Lenin and Trotsky (top), and Stalin and Nikolai Yezhov (bottom). Shown on the right are their altered counter parts where Trotsky and several others (top), and Yezhov (bottom) were removed.

works by spraying watercolor paint with a small pressure gun shaped like a pencil by means of compressed air. Re-touching is usually done by painting over the film with an extremely fine point brush, and requires the use of special dyes. Dodging and burning are manipulations that change the amount of exposure in a selected area of print: burning adds exposure, while dodging reduces it. Custom made photographic masks are employed when dodging or burning. The contrast and color of analog images are adjusted through the use of special light filters and photographic paper. All these manipulations require a high degree of technical expertise and sophisticated dark room equipment, which are, in general, outside the reach of the average user.

During the past few years affordable, high-resolution digital cameras have been rapidly replacing their film-based, analog counterparts. In addition to this, the advent of low-cost, high-performance computers, and sophisticated photo-editing and computer graphics software allow an average user to do complex image manipulations and create credible digital forgeries with relative ease. Although there are, perhaps, an uncountable number of ways to manipulate and tamper with digital images, we present here some of the most common types of digital image tampering [17]:

1. **Compositing**. This is one of the most common forms of digital tampering in which two, or more, digital images are spliced together to create a composite image. Shown in Figure 1.2 are an original image (left) and the same composited image (right). The composite image was created by overlaying the head of a different person (taken from an image not shown here) onto the shoulders of the original kayaker.

2. **Morphing**. This technique gradually transforms a source image into a target image. Shown

Figure 1.2: An original image (left) and a composite image (right), in which the head of another person was overlaid onto the shoulders of the original kayaker. The original photograph was downloaded from `www.freephoto.com`, and the composite photograph is from [17].

Figure 1.3: Example of a sequence of morphed images: a human face (the source) is slowly changed into an alien doll (the target) — morphed sequence from [17].

in Figure 1.3 is a sequence of images, in which a human face (the source) is being morphed into an alien doll (the target). Note how the shape and appearance of the source slowly change into the shape and appearance of the target. The intermediate images have features from both the source and target images, and have an aspect that is "part human, part alien".

3. **Re-touching**. This is a broad class of localized tampering techniques that include among others: air-brushing, smudging, blurring, painting, and copying and pasting of regions within the same image. Shown in Figure 1.4 is an original image of a person, and the same retouched image in which all previously mentioned techniques were employed. The tampering consisted in removing some facial hair and blemishes, smoothing the face, and whitening the teeth.

4. **Enhancing**. This is a class of more global techniques that include: color and contrast adjustment, blurring, and sharpening. Shown in Figure 1.5 are an original image, Figure 1.5(a), and the same image enhanced in three different ways: the color palette was altered such that the color of the blue police motorcycle was changed to cyan, and the red van in the background became yellow, Figure 1.5(b), the brightness was increased to make the image appear as if it was taken on a bright, sunny day, Figure 1.5(c), and the background was blurred obscuring the details of the parked cars, Figure 1.5(d). Although they don't fundamentally alter the appearance of the original, these manipulations may, however, have a subtle effect on the interpretation of an image. For example, they can alter the weather or the time of the day, or

Figure 1.4: An original image (left) and the same re-touched image (right). Some of the original person's facial hair was removed, his face was smoothed, and his teeth were whitened.

Figure 1.5: Shown are (a) an original image, and the same image enhanced by (b) altering the colors, (c) adjusting the contrast, and (d) blurring the background. The original photograph was downloaded from `www.freephoto.com`, and the enhanced photographs are from [17].

they can obscure some details while exaggerating others.

5. **Computer graphics**. In contrast to compositing, morphing, re-touching, and enhancing that work, in general, by altering actual photographs, this type of tampering refers to images that are generated using a computer and a graphics software package (e.g., *Maya*® by Alias®, or *3ds max*® by discreet®). Such images are generated by first constructing a 3-D polygonal model that embodies a desired shape. The model is then augmented with color and texture, and illuminated with one, or several, virtual light sources to approximate desired lighting conditions. Finally, the augmented model is rendered through a virtual camera to create a synthetic image. Shown in Figure 1.6 are two examples of model-based, computer-generated images of a person's bust. Good quality, computer-generated images that are hard to distinguish from real photographs generally require a talented and skilled computer artist. Emerging technologies that employ real people and laser scanners to acquire highly accurate three-dimensional models, may, however, make the creation of credible computer-

Figure 1.6: Two examples of model-based, computer-generated images of a person's bust. These images were created by Mihai Anghelescu, and downloaded from `www.xmg.ro/mihai/index.html`.

generated images easier and more accessible to less skilled people. A recent trend of interest in the context of digital forgeries, has been the creation of hybrid images that mix real and computer-generated images. For example, computer-generated images can be spliced into real photographs, and real images can be mapped directly onto a three-dimensional model, a technique known as image-based rendering.

6. **Painted**. This type of tampering refers to images created from a blank screen with a photo-editing or drawing software (e.g., *Adobe Photoshop*® or *Corel Draw*®) in a way similar to that of an artist painting on traditional canvas. This technique is the most challenging form of tampering, and would require a highly talented painter with good technical skills to yield realistic images.

Note that, when creating digital forgeries, the image manipulations presented above, and others, are often employed simultaneously. For example, two images may be spliced together, then the composite image may be re-touched and enhanced. Note also that tampering is not a well defined notion, and is often application dependent. Certain image manipulations may be legitimate in some cases, and illegitimate in others. For example, it may be acceptable to use a composite image for a magazine cover, but not as evidence in a court of law.

1.2 Watermarking

Watermarking has been defined [11] as "the practice of imperceptibly[2] altering a Work to embed a message about that Work", where *Work* refers to a specific image, audio clip, video clip, or other digital media. The basic approach employs an *embedder* that inserts an imperceptible watermark (e.g., a digital code, or a checksum) into the media, and a *decoder* that tries to determine if a watermark is present, and if so, outputs its encoded message.

Digital watermarking has many applications, including: broadcast monitoring, owner identification, proof of ownership, transaction tracking, content authentication, and copy and device control. Although not the only solution for these applications, digital watermarks are distinguished from other techniques by three defining characteristics: (1) they are imperceptible, (2) they are inseparable from the digital media they are embedded in, and (3) they undergo the same transformations as the digital media itself.

Watermarking systems can be characterized by a number of defining properties related to the embedding process (*effectiveness*, *fidelity*, and *data payload*), the detection process (*blind vs. informed*, *false positives*, and *robustness*), the *security* and use of *secret keys*, and the *cost* of embedding and detection. The embedding effectiveness is the probability of the watermark detection immediately after embedding. Although 100% effectiveness is desirable, it may not be always possible without compromising other desirable properties, e.g. fidelity. The fidelity of a watermarking scheme refers to the perceptual difference between the original and the watermarked media. Ideally, watermark embedding should not alter the perception of the cover media. Data payload is the number of bits that a watermark embeds in a unit of time or within a cover medium. Requirements for payload can vary from one bit (a watermark may be present or absent) to several thousand if, for example, error correction codes need to be embedded. Informed detection refers to watermarking schemes that require access to the original cover medium (or to some information derived from the original), while blind detection refers to schemes that do not need the original. Blind detection is employed by a majority of watermarking schemes. The false positive rate is the probability of detecting a watermark in a medium that does not contain one. The requirements for the false positive rate are application dependent and can vary from 10^{-6} for proof of ownership, to one in 10^{12} frames for DVD video. Robustness refers to the ability of a watermark to survive common signal processing operations, e.g., filtering, lossy compression, printing and scanning, and temporal and geometric distortions. In practice, watermarks are required to survive only a specific subset of signal processing operations. The security of a watermark refers to its capability to resist hostile attacks. Attacks on watermarking schemes typically consist of unauthorized removal, embedding, or detection. The unauthorized removal refers to active attacks that prevent the detectability of a watermark either by masking or by eliminating it. The unauthorized embedding, also known as forgery, refers to the capability of an adversary to insert a watermark into a medium with the goal to falsely authenticate it. Unauthorized detection, a passive attack, is

[2]Imperceptibility is not considered by some researchers as a defi ning characteristic of digital watermarks, and perceptible watermarking is an active area of research (e.g., [44]). Perceptible watermarks, while useful in conveying an immediate claim of ownership, eliminate the potential commercial value of the media they are embedded in, which restricts their area of applicability.

of concern in information hiding applications, e.g., steganography. The security of watermarking can be enhanced by the use of secret keys: for example, a key is employed for embedding, and a matching key is required for detection. Another important issue watermarking schemes need to contend with is the computational cost of the embedding and detection. The cost requirements vary from real-time for broadcast monitoring to "as long as necessary" for content authentication and proof of ownership. Depending on the intended application, watermarks may have different, and sometimes conflicting, required properties. For example, a digital watermark intended for owner identification needs to be robust, while fragility may be a desired property for watermarks intended for content authentication.

One of the most important applications of digital watermarks is the authentication of digital images (see, for example, [12, 31, 11] for general surveys). In this context, watermarking techniques revolve around answering the following questions:

- Has a digital image been altered at all?
- Has a digital image been significantly altered so that to change its meaning?
- If a digital image has been altered, what portions have been tampered with?
- Can a tampered digital image be restored?

Exact authentication, the most basic task, consists in verifying if an image has been altered at all since it left a trusted party, that is, changing even a single image bit is considered unacceptable. Approaches proposed for exact authentication include *fragile watermarks* [63], *embedded signatures* [25, 60], and *erasable watermarks* [30, 23]. Fragile watermarks are simply watermarks designed to become undetectable when the image is changed in any way. An example of such a watermark is to embed a predefined bit sequence in the least significant bit (LSB) of an image. Some fragile watermarking systems may have embedding methods that depend on a particular media format: for example, watermarks may be embedded in the block discrete cosine transform (DCT) coefficients of JPEG compressed images, or in different types of MPEG encoded video frames. The method of embedded signatures works by embedding at the time of recording an authentication signature in an image. This signature can be embedded using either a robust or a fragile watermark, the latter being typically preferred because it is simpler to implement. Erasable watermarks, also known as *invertible*, are designed such that their removal leaves an exact copy of the cover image. They are employed in applications that do not tolerate the slight image changes that incur when inserting and extracting a typical watermark.

Selective authentication refers to verifying if an image has been modified by any illegitimate distortions. Whether a distortion is illegitimate or not depends on the intended application. Proposed approaches include *semi-fragile watermarks* [37] and *signatures* [56, 58, 41, 5], and *tell-tale* watermarks [35, 65]. The semi-fragile watermarks are designed to survive only legitimate distortions: for example, Lin and Chang [37] proposed an image authentication method that is relatively insensitive to lossy JPEG compression. Semi-fragile signatures work like their fragile counterparts, but are computed from image properties that are unlikely to be changed by legitimate distortions,

e.g., perceptually significant components such as the low frequency block DCT coefficients. Tell-tale watermarks are robust watermarks that survive tampering, but are distorted in the process. The analysis of a distorted tell-tale watermark can then reveal the tampering and distortion type.

Many watermarking systems are designed with the ability to identify not only if an image was altered, but to localize the tampering as well, a capability referred to as *localization*. Most localization approaches rely on some form of block-wise [61, 21, 8], or pixel-wise [64] authentication. For example, Yeung and Mintzer [64] have proposed a method that employs a pseudo-random mapping from pixel intensities to binary values. The encoder changes the pixel intensities such that the pseudo-random mapping would yield a desired binary pattern, e.g., a tiled, binary image in the form of a logo. In addition to block and pixel-wise authentication, tell-tale watermarks may also have the localization capability.

Restoration refers to the ability of a watermarking scheme to restore image regions that have been corrupted. One possible approach is to embed redundant information in the image, e.g., error detection and correction codes. Although this approach allows perfect restoration, it is, however, impractical as the amount of information to be embedded is significant. More practical approaches, self-embedding [22, 38] and blind restoration [33, 34], allow only approximate restoration. Self-embedding works by embedding a highly compressed version of an image into itself: for example, a low quality, lossy compressed version of a given image can be embedded into the least significant bits of its own pixels. Blind restoration employs a tell-tale watermark to determine what distortions have been applied to an image, then attempts to invert those distortions.

A recent approach to implement a secure digital camera has been proposed by Blythe and Fridrich [7]. The method works by first collecting biometric data of the camera user, cryptographic hashes of the camera scene, the date and time, and other forensic data. This information is then inserted into the image using an invisible, *erasable* watermark (the image is left intact after the watermark extraction). The whole procedure links the photographer to the photograph, in addition to providing protection to tampering, and may prove to be useful for investigations by law enforcement examiners.

A major drawback of watermarking-based approaches is that the watermarks must be inserted either at the time of recording, or afterward by a person authorized to do so, which requires specially equipped cameras or subsequent processing of the original image. These methods also rely on the assumption that it is difficult to remove and reinsert digital watermarks in images. It is unclear if this is a reasonable assumption (e.g., [13]).

1.3 Related Work

Detecting digital tampering in the absence of watermarks or signatures is a relatively new research topic, and only few results have been published so far. Farid [16] proposed a technique for authenticating digital audio signals. This technique works under the assumption that natural signals have low higher-order correlations in the frequency domain. He then showed that higher-order correlations are introduced in the frequency domain when a natural audio signal undergoes a non-linear transformation, as would likely happen in the process of creating a digital forgery. Tools

from polyspectral analysis, i.e., the bispectrum, were applied to quantify and detect these correlations. This approach was extended by Ng and Chang [47] to detect splicing in composite images. Farid and Popescu have also applied this technique to blindly detect the presence of multiple non-linearities in digital images [51], which represents a possible sign of tampering.

Lyu and Farid [40] have proposed a method to distinguish between photographic and computer-generated, photo-realistic images. Their method works by representing an image with a multi-scale, multi-resolution, wavelet-like transform, then extracting a feature vector of first and higher-order statistics from the transform coefficients. Statistical learning tools (e.g., linear discriminant analysis and non-linear support vector machines) are employed on these feature vectors to discriminate between photographic and photo-realistic images.

Fridrich *et al.* [24] have proposed a method to detect duplicated regions in digital images. This method, similar to the one we propose in Chapter 5, works by lexicographically sorting the DCT coefficients of fixed-size image blocks, and examining neighboring blocks in the sorted order. Similar neighboring blocks are then employed to highlight the potential duplicated regions in the image.

1.4 Contributions

We describe a set of statistical techniques for detecting traces of tampering in digital images. These techniques work in the complete absence of any digital watermark or signature. The common theme, and unifying framework, of these techniques is the assumption that digital forgeries, often visually imperceptible, alter some underlying statistical properties of natural images. Within this framework, our approach consists in first identifying the statistical changes associated with particular kinds of tampering, then designing techniques for estimating these changes. Following this approach, we have developed five techniques that quantify and detect different types of tampering:

1. **Re-sampled Images**. Imagine a forgery created by splicing together images of two people and overlaying them on a chosen background. In order to create a convincing match it is often necessary to re-size, rotate, or stretch the original images (or portions of them), which requires re-sampling the images onto a different lattice. Re-sampling introduces specific correlations between neighboring image pixels, that are quantified and detected using the expectation-maximization (EM) algorithm.

2. **Manipulated Color Filter Array (CFA) Interpolated Images**. Most digital cameras are equipped with a single charge-coupled device (CCD) or complementary metal oxide semiconductor (CMOS) sensor, and capture color images using an array of color filters. At each pixel location, only a single color sample (out of three) is captured. The missing color samples are then inferred from neighboring values. This process, known as CFA interpolation or demosaicking, introduces specific correlations between the samples of a color image. These correlations are typically destroyed when a CFA interpolated image is tampered with, and can be employed to uncover traces of tampering. Using an approach similar to the re-sampling detection, we have employed the EM algorithm to detect if the CFA interpolation correlations are missing in any portion of an image.

3. **Double JPEG Compression**. When tampering with an image, a typical pattern is to load the image into a photo-editing software (e.g., *Adobe Photoshop*®), do some processing, and re-save the tampered image. If JPEG format is used to store the images, the resulting tampered image has been double compressed. Double JPEG compression introduces specific correlations between the discrete cosine transform (DCT) coefficients of image blocks. These correlations can be detected and quantified by examining the histograms of the DCT coefficients. While double JPEG compression of an image does not necessarily prove malicious tampering, it raises suspicions that the image may not be authentic.

4. **Duplicated Image Regions**. A common manipulation in removing an unwanted person or object from an image, is to copy and paste portions of the same image over the desired region. If the splicing is imperceptible, little concern is typically given to the fact that identical regions are present in the image. We have developed an algorithm that employs a principal component analysis (PCA) on fixed-size image blocks, and lexicographic sorting to efficiently detect the presence of duplicated regions even in noisy or lossy compressed images.

5. **Inconsistent Noise Patterns**. Digital images contain an inherent amount of noise that is largely uniformly distributed across an entire image. When creating digital forgeries, it is common to add small amounts of localized noise to tampered regions in order to conceal traces of tampering (e.g., at a splice boundary). As a result, local noise levels across the image may become inconsistent. We have implemented an algorithm for blind estimation of localized noise variance, and employed it to differentiate regions with different amounts of additive noise.

For each of the above techniques, we develop its theoretical foundation, show its effectiveness in detecting credible forgeries, and analyze its sensitivity and robustness to simple counter-attacks.

Chapter 2

Re-sampled Images

Consider the scenario in which a digital forgery is created by splicing together two or more images. In order to create a convincing match, it is often necessary to re-size, rotate, or stretch the images, or portions of them. This procedure typically requires re-sampling an image onto a new sampling lattice using an interpolation technique (e.g., bi-cubic). Although a re-sampled image is often imperceptible, it contains specific correlations that, when detected, may represent evidence of tampering. We describe the form of these correlations, and propose an algorithm for detecting them in any portion of an image.

For purposes of exposition we will first describe how and where re-sampling introduces correlations in 1-D signals, and how to detect these correlations. The relatively straightforward generalization to 2-D images is then presented. To illustrate the general effectiveness of this technique, we show results on re-sampled natural test images and on perceptually credible forgeries. Also presented is an extensive set of experiments that test the sensitivity of re-sampling detection for a large set of parameters, as well as the technique's robustness to simple counter-attacks and lossy image compression schemes.

2.1 Re-sampling Signals

Consider a 1-D discretely-sampled signal $x[t]$ with m samples, Figure 2.1(a). The number of samples in this signal can be increased or decreased by a factor p/q to n samples in three steps [48]:

1. **up-sample**: create a new signal $x_u[t]$ with pm samples, where $x_u[pt] = x[t]$, $t = 1, 2, ..., m$, and $x_u[t] = 0$ otherwise, Figure 2.1(b).

2. **interpolate**: convolve $x_u[t]$ with a low-pass filter: $x_i[t] = x_u[t] \star h[t]$, Figure 2.1(c).

3. **down-sample**: create a new signal $x_d[t]$ with n samples, where $x_d[t] = x_i[qt]$, $t = 1, 2, ..., n$. Denote the re-sampled signal as $y[t] \equiv x_d[t]$, Figure 2.1(d).

Different types of re-sampling algorithms (e.g., linear, cubic [32]) differ in the form of the interpolation filter $h[t]$ in step 2.

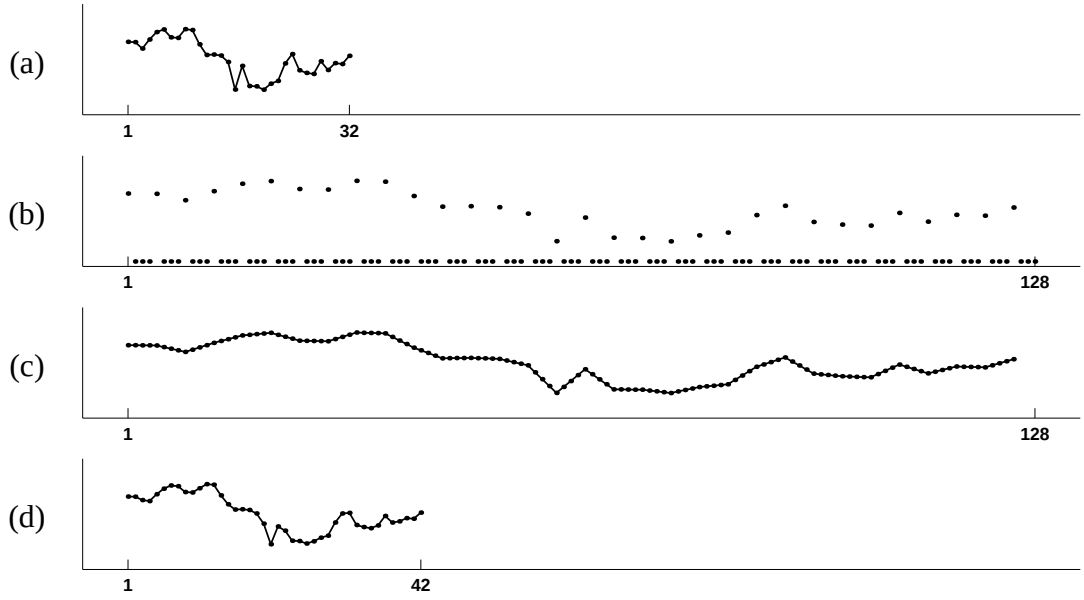

Figure 2.1: Re-sampling a signal by a factor of $4/3$: shown are (a) the original signal; (b) the up-sampled signal; (c) the interpolated signal; and (d) the final re-sampled signal.

Since all three steps in the re-sampling of a signal are linear, this process can be described with a matrix-based, single linear equation. Denoting the original and re-sampled signals in vector form, $\vec{x}$ and $\vec{y}$, respectively, re-sampling takes the form:

$$\vec{y} \;=\; A_{p/q}\vec{x}, \tag{2.1}$$

where the $n \times m$ matrix $A_{p/q}$ embodies the entire re-sampling process. For example, the matrix for up-sampling by a factor of $4/3$ using linear interpolation (Figure 2.1) has the form:

$$A_{4/3} \;=\; \begin{bmatrix} 1 & 0 & 0 & 0 & \\ 0.25 & 0.75 & 0 & 0 & \\ 0 & 0.50 & 0.50 & 0 & \\ 0 & 0 & 0.75 & 0.25 & \\ 0 & 0 & 0 & 1 & \\ & & & & \ddots \end{bmatrix}. \tag{2.2}$$

Depending on the re-sampling rate, the re-sampling process will introduce correlations of varying degrees between neighboring samples. For example, consider the up-sampling of a signal by a factor of two using linear interpolation. In this case, the re-sampling matrix takes the form:

$$A_{2/1} = \begin{bmatrix} 1 & 0 & 0 & \\ 0.5 & 0.5 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0.5 & 0.5 & \\ 0 & 0 & 1 & \\ & & & \ddots \end{bmatrix}. \tag{2.3}$$

Here, the odd samples of the re-sampled signal $\vec{y}$ take on the values of the original signal $\vec{x}$, i.e., $y_{2i-1} = x_i$, $i = 1, \ldots, m$. The even samples, on the other hand, are the average of adjacent neighbors of the original signal:

$$y_{2i} = 0.5x_i + 0.5x_{i+1}, \tag{2.4}$$

where $i = 1, \ldots, m-1$. Note that since each sample of the original signal can be found in the re-sampled signal, i.e., $x_i = y_{2i-1}$ and $x_{i+1} = y_{2i+1}$, the above relationship can be expressed in terms of the re-sampled samples only:

$$y_{2i} = 0.5y_{2i-1} + 0.5y_{2i+1}. \tag{2.5}$$

That is, across the entire re-sampled signal, each even sample is precisely the same linear combination of its two adjacent neighbors. In this simple case, at least, a re-sampled signal could be detected (in the absence of noise) by noticing that every other sample is perfectly correlated to its neighbors. To be useful in a general forensic setting we need, at a minimum, for these types of correlations to be present regardless of the re-sampling rate.

Consider now re-sampling a signal by an arbitrary amount p/q. In this case we first ask, when is the i^{th} sample of a re-sampled signal equal to a linear combination of its $2N$ neighbors, that is:

$$y_i \stackrel{?}{=} \sum_{k=-N}^{N} \alpha_k y_{i+k}, \tag{2.6}$$

where α_k are scalar weights (and $\alpha_0 = 0$). Re-ordering terms, and re-writing the above constraint in terms of the re-sampling matrix yields:

$$y_i - \sum_{k=-N}^{N} \alpha_k y_{i+k} = 0 \tag{2.7}$$

$$(\vec{a}_i \cdot \vec{x}) - \sum_{k=-N}^{N} \alpha_k (\vec{a}_{i+k} \cdot \vec{x}) = 0 \tag{2.8}$$

$$\left(\vec{a}_i - \sum_{k=-N}^{N} \alpha_k \vec{a}_{i+k} \right) \cdot \vec{x} = 0, \tag{2.9}$$

where $\vec{a}_i$ is the i^{th} row of the re-sampling matrix $A_{p/q}$, and $\vec{x}$ is the original signal. We see now that the i^{th} sample of a re-sampled signal is equal to a linear combination of its neighbors when the i^{th} row of the re-sampling matrix, $\vec{a}_i$, is equal to a linear combination of the neighboring rows, $\sum_{k=-N}^{N} \alpha_k \vec{a}_{i+k}$. For example, in the case of up-sampling by a factor of two, Equation (2.3), the even rows are a linear combination of the two adjacent odd rows. Note also that if the i^{th} sample is a linear combination of its neighbors then the $(i - kp)^{th}$ sample (k an integer) will be the same combination of its neighbors, that is, the correlations are periodic. It is, of course, possible for the constraint of Equation (2.9) to be satisfied when the difference on the left-hand side of the equation is orthogonal to the original signal $\vec{x}$. While this may occur on occasion, these correlations are unlikely to be periodic.

2.2 Detecting Re-sampling

Given a signal that has been re-sampled by a known amount and interpolation method, it is possible to find a set of periodic samples that are correlated in the same way to their neighbors. For example, consider again the re-sampling matrix of Equation (2.2). Here, based on the periodicity of the re-sampling matrix, we see that, for example, the 3^{rd}, 7^{th}, 11^{th}, etc. samples of the re-sampled signal will have the same correlations to their neighbors. The specific form of the correlations can be determined by finding the neighborhood size, N, and the set of coefficients, $\vec{\alpha}$, that satisfy: $\vec{a}_i = \sum_{k=-N}^{N} \alpha_k \vec{a}_{i+k}$, Equation (2.9), where $\vec{a}_i$ is the i^{th} row of the re-sampling matrix and $i = 3, 7, 11$, etc. If, on the other-hand, we know the specific form of the correlations, $\vec{\alpha}$, then it is straightforward to determine which samples satisfy $y_i = \sum_{k=-N}^{N} \alpha_k y_{i+k}$, Equation (2.7).

In practice, of course, neither the samples that are correlated, nor the specific form of the correlations are known. In order to determine if a signal has been re-sampled, we employ the expectation-maximization algorithm[1] (EM) [14] to simultaneously estimate a set of periodic samples that are correlated to their neighbors, and the specific form of these correlations. We begin by assuming that each sample belongs to one of two models. The first model, M_1, corresponds to those samples y_i that are correlated to their neighbors, i.e., are generated according to the following model:

$$M_1 : y_i = \sum_{k=-N}^{N} \alpha_k y_{i+k} + n(i), \tag{2.10}$$

where the model parameters are given by the specific form of the correlations, $\vec{\alpha}$ ($\alpha_0 = 0$), and $n(i)$ denote independently, and identically distributed (i.i.d.) samples drawn from a Gaussian distribution with zero mean and unknown variance σ^2. The second model, M_2, corresponds to those samples that are not correlated to their neighbors, i.e., are generated by an outlier process. The EM algorithm is a two-step iterative algorithm: (1) in the E-step the probability that each sample belongs to each model is estimated; and (2) in the M-step the specific form of the correlations between samples is estimated. More specifically, in the E-step, the probability of each sample y_i

[1] A short tutorial on the general EM algorithm, and an example of its application to a mixture models problem are presented in Appendix A.

belonging to model M_1 can be obtained using Bayes' rule:

$$\Pr\{y_i \in M_1 \mid y_i\} \;=\; \frac{\Pr\{y_i \mid y_i \in M_1\}\Pr\{y_i \in M_1\}}{\sum_{k=1}^{2}\Pr\{y_i \mid y_i \in M_k\}\Pr\{y_i \in M_k\}}, \tag{2.11}$$

where the priors $\Pr\{y_i \in M_1\}$ and $\Pr\{y_i \in M_2\}$ are assumed to be equal to $1/2$. The probability of observing a sample y_i knowing it was generated by model M_1 is given by:

$$\Pr\{y_i \mid y_i \in M_1\} \;=\; \frac{1}{\sigma\sqrt{2\pi}}\exp\left[\frac{-\left(y_i - \sum_{k=-N}^{N}\alpha_k y_{i+k}\right)^2}{2\sigma^2}\right]. \tag{2.12}$$

We assume that the probability of observing samples generated by the outlier model, $\Pr\{y_i \mid y_i \in M_2\}$, is uniformly distributed over the range of possible values of y_i. The variance, σ^2, of the Gaussian distribution in Equation (2.12) is estimated in the M-step (see Section 2.6). Note that the E-step requires an estimate of $\vec{\alpha}$, which on the first iteration is chosen randomly. In the M-step, a new estimate of $\vec{\alpha}$ is computed using weighted least-squares, that is, minimizing the following quadratic error function:

$$E(\vec{\alpha}) \;=\; \sum_i w(i)\left(y_i - \sum_{k=-N}^{N}\alpha_k y_{i+k}\right)^2, \tag{2.13}$$

where the weights $w(i) \equiv \Pr\{y_i \in M_1 \mid y_i\}$, Equation (2.11), and $\alpha_0 = 0$. This error function is minimized by computing the gradient with respect to $\vec{\alpha}$, and setting the result equal to zero. Solving for $\vec{\alpha}$ yields:

$$\vec{\alpha} \;=\; (Y^T W Y)^{-1} Y^T W \vec{y}, \tag{2.14}$$

where the matrix Y is:

$$Y \;=\; \begin{bmatrix} y_1 & \cdots & y_N & y_{N+2} & \cdots & y_{2N+1} \\ y_2 & \cdots & y_{N+1} & y_{N+3} & \cdots & y_{2N+2} \\ \vdots & & \vdots & \vdots & & \vdots \\ y_i & \cdots & y_{N+i-1} & y_{N+i+1} & \cdots & y_{2N+i} \\ \vdots & & \vdots & \vdots & & \vdots \end{bmatrix}, \tag{2.15}$$

and W is a diagonal weighting matrix with $w(i)$ along the diagonal.

The E-step and the M-step are iteratively executed until a stable estimate of $\vec{\alpha}$ is achieved — see Section 2.6 for a detailed algorithm. The final $\vec{\alpha}$ has the property that it maximizes the likelihood of the observed samples[2] — see Appendix A for more details on maximum likelihood estimation and EM.

[2]The EM algorithm is proved to always converge to a stable estimate that is a local maximum of the log-likelihood of the observed data, [14] and Appendix A.

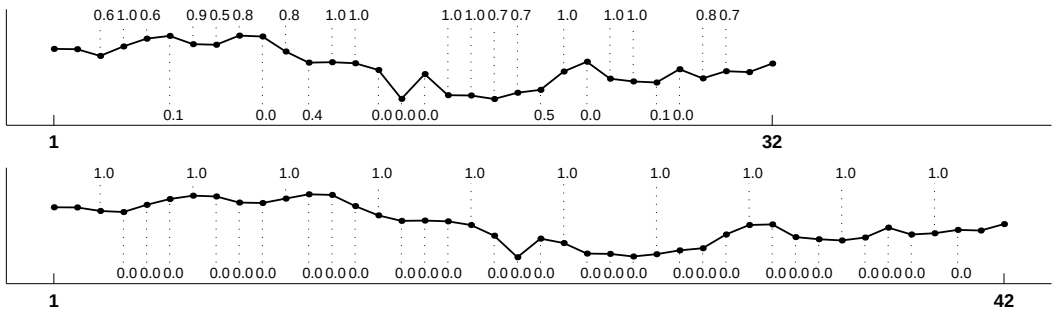

Figure 2.2: A signal with 32 samples (top) and this signal re-sampled by a factor of $4/3$ using cubic interpolation (bottom). Each sample is annotated with its probability of being correlated to its neighbors. Note that for the up-sampled signal these probabilities are periodic, while for the original signal they are not.

Shown in Figure 2.2 are the results of running EM on the original and re-sampled signals of Figure 2.1. Shown on top is the original signal where each sample is annotated with its probability of being correlated to its neighbors (the first and last two samples are not annotated due to border effects — a neighborhood size of five, $N = 2$, was used in this example). Similarly, shown on the bottom is the re-sampled signal and the corresponding probabilities. In the latter case, the periodic pattern is obvious: only every 4^{th} sample has probability 1, as would be expected from an up-sampling by a factor of $4/3$, Equation (2.2). As expected, no periodic pattern is present in the original signal. The periodic pattern introduced by re-sampling depends, of course, on the re-sampling rate. As a result, it is possible to not only uncover traces of re-sampling, but to also estimate the amount of re-sampling. There are, however, parameters that produce indistinguishable periodic patterns[3], which means that the amount of re-sampling can be estimated only within an inherent ambiguity. Since we are primarily concerned with detecting traces of re-sampling, and not necessarily the amount of re-sampling, this limitation is not critical.

There is a range of re-sampling rates that will not introduce periodic correlations. For example, consider down-sampling by a factor of two (for simplicity, consider the case where there is no interpolation). The re-sampling matrix, in this case, is given by:

$$A_{1/2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 1 & \\ & & & & & \ddots \end{bmatrix}. \tag{2.16}$$

[3]It can be shown that given two re-sampling rates p/q and r/s (p, q, r, s integers), they will produce identical patterns if:

1. $p/q - r/s$ has an integer value, or
2. $\{p/q\} + \{r/s\} = 1$, where $\{\cdot\}$ denotes the fractional part of a number.

For example, $1/4$, $3/4$, and $5/4$ will produce identical patterns.

Note that no row can be written as a linear combination of neighboring rows (the rows are orthogonal to one another), and that, in this case, re-sampling is not detectable. In general, the detectability of any re-sampling can be determined by generating the re-sampling matrix, and checking whether any rows can be expressed as a linear combination of neighboring rows.

2.3 Re-sampling Images

In the previous sections we showed that for 1-D signals re-sampling introduces periodic correlations, and that these correlations can be detected using the EM algorithm. The extension to 2-D images is relatively straightforward. As with 1-D signals, the up-sampling or down-sampling, of an image is still linear and involves the same three steps: up-sampling, interpolation, and down-sampling — these steps are simply carried out on a 2-D lattice. Again, as with 1-D signals, the re-sampling of an image introduces periodic correlations. Though we will only show this for up- and down-sampling, the same is true for an arbitrary affine transform[4].

Consider, for example, the simple case of up-sampling by a factor of two. Shown in Figure 2.3 are, from left to right, a portion of an original 2-D sampling lattice, the same lattice up-sampled by a factor of two, and a subset of the pixels of the re-sampled image. Assuming linear interpolation, these pixels are given by:

$$\begin{aligned} y_2 &= 0.5y_1 + 0.5y_3 \\ y_4 &= 0.5y_1 + 0.5y_7 \\ y_5 &= 0.25y_1 + 0.25y_3 + 0.25y_7 + 0.25y_9, \end{aligned} \tag{2.17}$$

and where $y_1 = x_1$, $y_3 = x_2$, $y_7 = x_3$, $y_9 = x_4$. Note that the pixels of the re-sampled image in the odd rows and even columns (e.g., y_2) will all be the same linear combination of their two horizontal neighbors. Similarly, the pixels of the re-sampled image in the even rows and odd columns (e.g., y_4) will all be the same linear combination of their two vertical neighbors. That is, as with 1-D signals, the correlations due to re-sampling are periodic. A 2-D version of the EM algorithm can then be applied to uncover these correlations in images.

2.4 Results

For the results presented here, we employed approximately 200 gray-scale images and 50 color images in TIFF format, both 512×512 pixels in size. Each of these images were cropped from a smaller set of thirty-five 1200×1600 images taken with a Nikon Coolpix 950 camera (the camera was set to capture and store in uncompressed TIFF format). Using bi-cubic interpolation these images were up-sampled, down-sampled, rotated, or affine transformed by varying amounts.

For the original and re-sampled images, the EM algorithm described in Section 2.2 was used to estimate probability maps that embody the correlations between pixels and their neighbors. The

[4]Even non-linear transformations introduce detectable correlations, but these correlations may not be periodic.

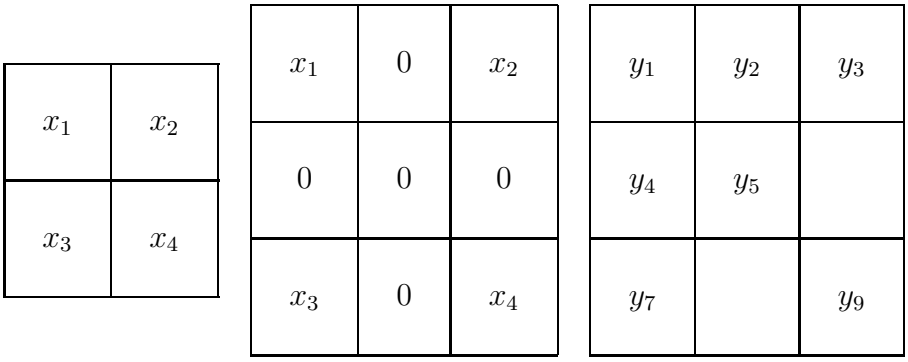

Figure 2.3: Shown from left to right are: a portion of the 2D lattice of an image, the same lattice up-sampled by a factor of 2, and a portion of the lattice of the re-sampled image.

EM parameters were fixed throughout at $N = 2$, $\sigma_0 = 0.0075$, $N_h = 3$ [5], and $p_0 = 1/256$ (see Section 2.6). Shown in Figures 2.4-2.6 are several examples of the periodic patterns that emerged due to re-sampling. In the top row of each figure are (from left to right) the original image, the estimated probability map and the magnitude of the central portion of the Fourier transform of this map (for display purposes, each Fourier transform was independently auto-scaled to fill the full intensity range and high-pass filtered to remove the lowest frequencies). Shown below this row is the same image uniformly re-sampled at different rates. For the re-sampled images, note the periodic nature of their probability maps and the strong, localized peaks in their corresponding Fourier transforms. Shown in Figure 2.7 are examples of the periodic patterns that emerge from four different affine transforms.

Shown in Figure 2.8 are results from applying consecutive re-samplings. Specifically, the image in the left column is the result of up-sampling by 15% an original image, then rotating by 5° the up-sampled image. The same operations were performed in reverse order on the same original image to yield the image in the right column. Note that while the images are perceptually indistinguishable, the periodic patterns that emerge are quite distinct, with the last re-sampling operation dominating the pattern. Note also that the corresponding Fourier transforms contain several sets of peaks corresponding to both re-sampling operations. As with a single re-sampling, consecutive re-samplings can be easily detected.

Shown in Figures 2.9-2.11 are several examples of our detection algorithm applied to images where only a portion of the image was re-sampled. Regions in each image were subjected to a range of stretching, rotation, shearing, etc. (these manipulations were done in *Adobe Photoshop*®). Shown in each figure is the original photograph, the forgery (where a portion of the original is replaced or altered), and the estimated probability map. Note that in each case, the re-sampled region is clearly detected — while the periodic patterns are not particularly visible in the spatial domain at the reduced scale, the well localized peaks in the Fourier domain clearly reveal their presence (for display purposes, each Fourier transform was independently auto-scaled to fill the

[5] The blurring of the residual error with a binomial filter of size $N_h \times N_h$ is not critical, but merely accelerates the convergence.

full intensity range and high-pass filtered to suppress the lowest frequencies). Note also that in Figure 2.9 the white sheet of paper on top of the trunk has strong activation in the probability map — when seen in the Fourier domain, however, it is clear that this region is not periodic, but rather uniform, and thus not representative of a re-sampled region.

2.4.1 Sensitivity and Robustness

From a digital forensic perspective it is important to quantify the robustness and sensitivity of our detection algorithm. To this end, it is first necessary to devise a quantitative measure of the amount of periodicity found in the estimated probability maps. To do so, we compare the estimated probability map with a set of synthetically generated probability maps that contain periodic patterns similar to those that emerge from re-sampled images.

Quantifying Re-sampling Correlations

Given a set of re-sampling parameters and interpolation method, a synthetic map is generated based on the periodicity of the re-sampling matrix. There are, however, several possible periodic patterns that may emerge in a re-sampled image. For example, in the case of up-sampling by a factor of two using linear interpolation, Equation (2.17), the coefficients $\vec{\alpha}$ estimated by the EM algorithm (with a 3×3 neighborhood) are expected to be one of the following:

$$\vec{\alpha}_1 = \begin{bmatrix} 0 & 0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix} \quad \vec{\alpha}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix} \quad \vec{\alpha}_3 = \begin{bmatrix} 0.25 & 0 & 0.25 \\ 0 & 0 & 0 \\ 0.25 & 0 & 0.25 \end{bmatrix}. \tag{2.18}$$

We have observed that EM will return one of these estimates only when the initial value of $\vec{\alpha}$ is close to one of the above three values, the neighborhood size is 3, and the initial variance of the conditional probability (σ in Equation (2.12)) is relatively small. In general, however, this fine tuning of the starting conditions is not practical. To be broadly applicable, we randomly choose an initial value for $\vec{\alpha}$, and set the neighborhood size and initial value of σ to values that afford convergence for a broad range of re-sampling parameters. Under these conditions, we have found that for specific re-sampling parameters and interpolation method, the EM algorithm typically converges to a unique set of linear coefficients. In the above example of up-sampling by a factor of two the EM algorithm typically converges to:

$$\vec{\alpha} = \begin{bmatrix} -0.25 & 0.5 & -0.25 \\ 0.5 & 0 & 0.5 \\ -0.25 & 0.5 & -0.25 \end{bmatrix}. \tag{2.19}$$

Note that this solution is different than each of the solutions in Equation (2.18). Yet, the relationships in Equation (2.17) are still satisfied by this choice of coefficients. Since the EM algorithm typically converges to a unique set of linear coefficients, there is also a unique periodic pattern that emerges. It is possible to predict this pattern by analyzing the periodic patterns that emerge from a large set of images. In practice, however, this approach is computationally demanding,

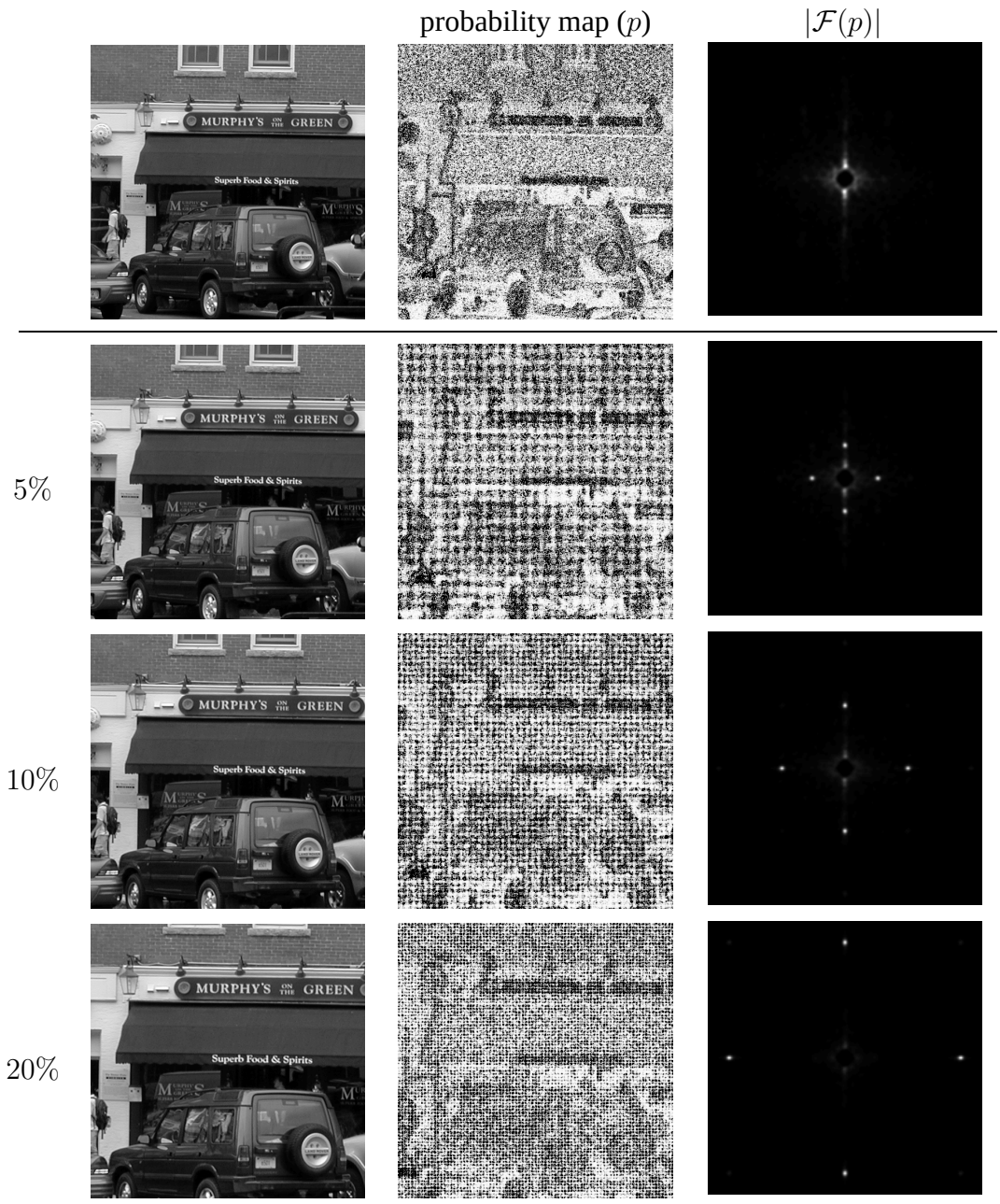

Figure 2.4: Shown in the top row is the original image, and shown below is the same image up-sampled by varying amounts. Shown in the middle column are the estimated probability maps (p) that embody the spatial correlations in the image. The magnitude of the Fourier transform of each map is shown in the right-most column. Note that only the re-sampled images yield periodic maps.

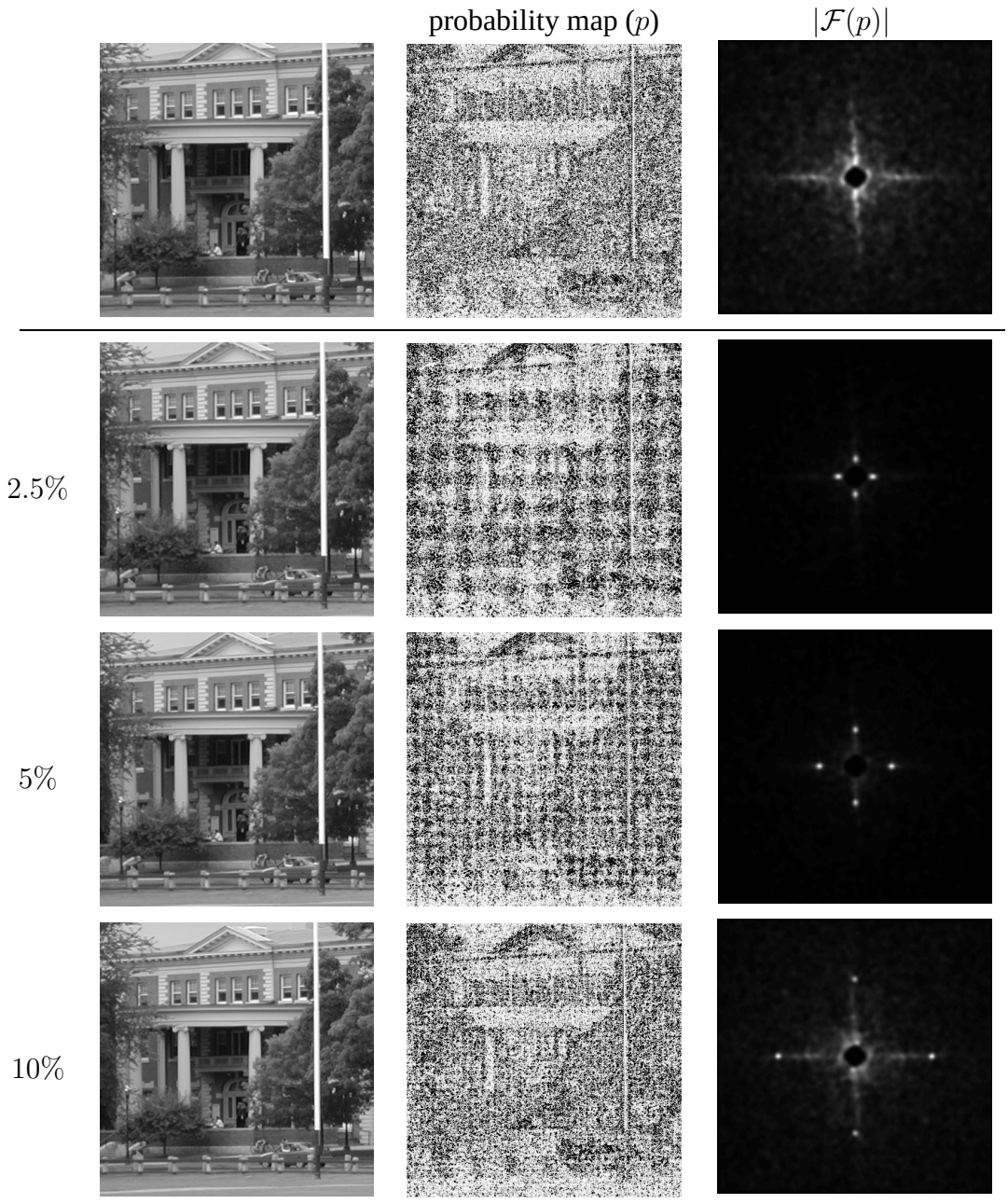

Figure 2.5: Shown in the top row is the original image, and shown below is the same image down-sampled by varying amounts. Shown in the middle column are the estimated probability maps (p) that embody the spatial correlations in the image. The magnitude of the Fourier transform of each map is shown in the right-most column. Note that only the re-sampled images yield periodic maps.

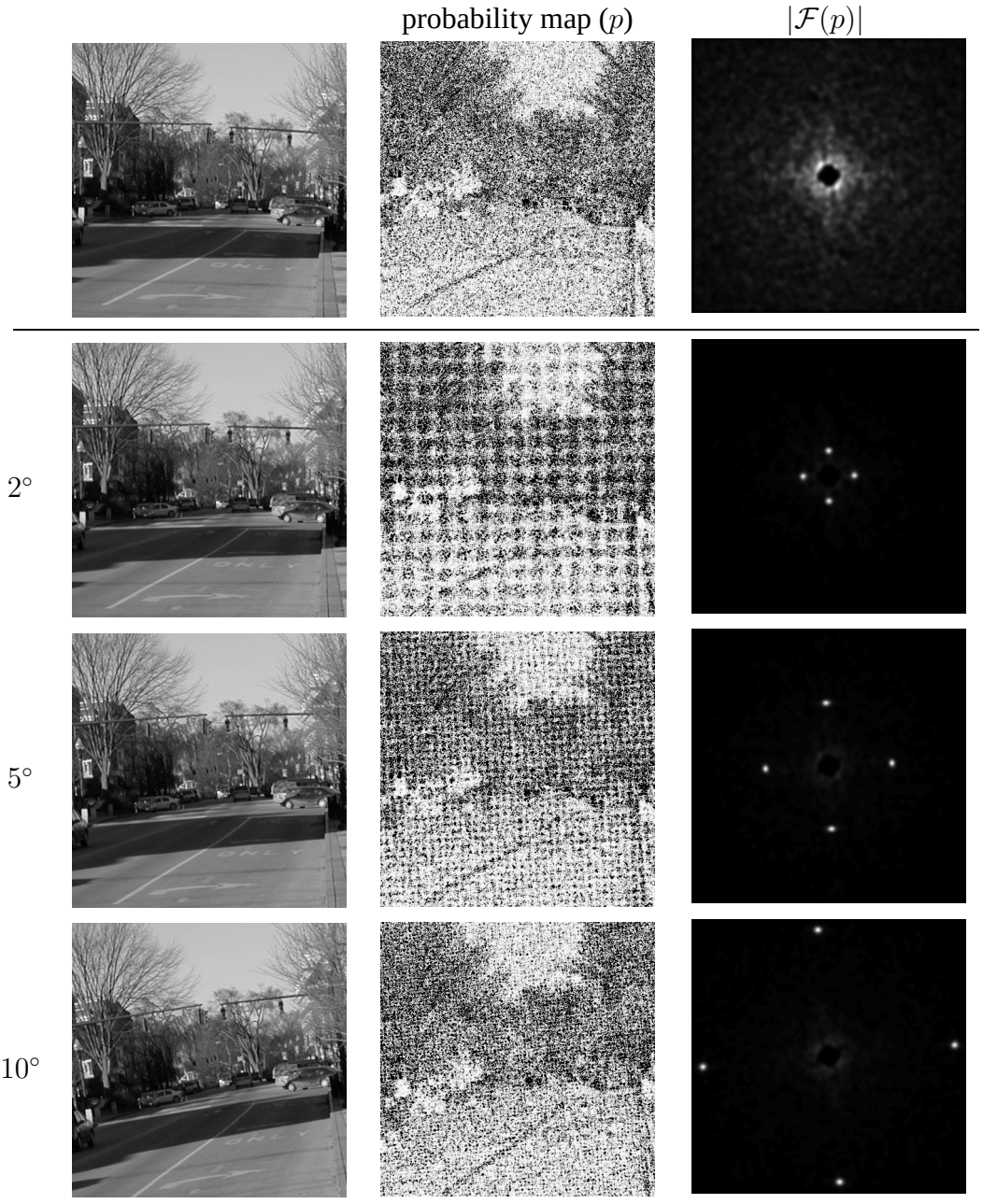

Figure 2.6: Shown in the top row is the original image, and shown below is the same image rotated by varying amounts. Shown in the middle column are the estimated probability maps (p) that embody the spatial correlations in the image. The magnitude of the Fourier transform of each map is shown in the right-most column. Note that only the re-sampled images yield periodic maps.

Figure 2.7: Shown in the left column are four images affine transformed by random amounts. Shown in the middle column are the estimated probability maps (p) that embody the spatial correlations in the image. The magnitude of the Fourier transform of each map is shown in the right-most column. Note that these images yield periodic maps.

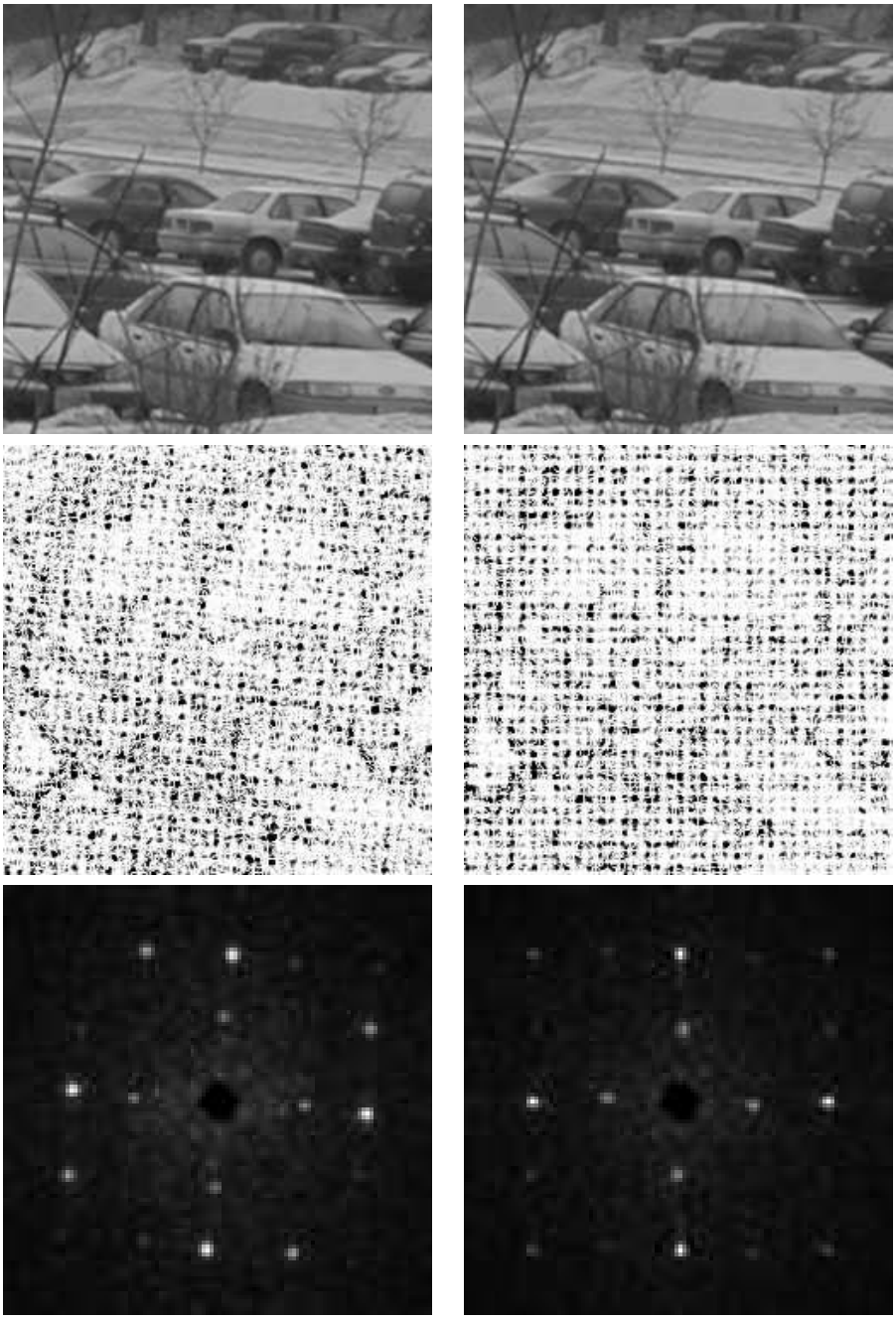

Figure 2.8: Shown are two images that were consecutively re-sampled (top left: up-sampled by 15% and then rotated by $5°$; top right : rotated by $5°$ and then up-sampled by 15%). Shown in the second row are the estimated probability maps that embody the spatial correlations in the image. The magnitude of the Fourier transform of each map is shown in the bottom column — note the presence of multiple peaks that correspond to both the rotation and up-sampling.

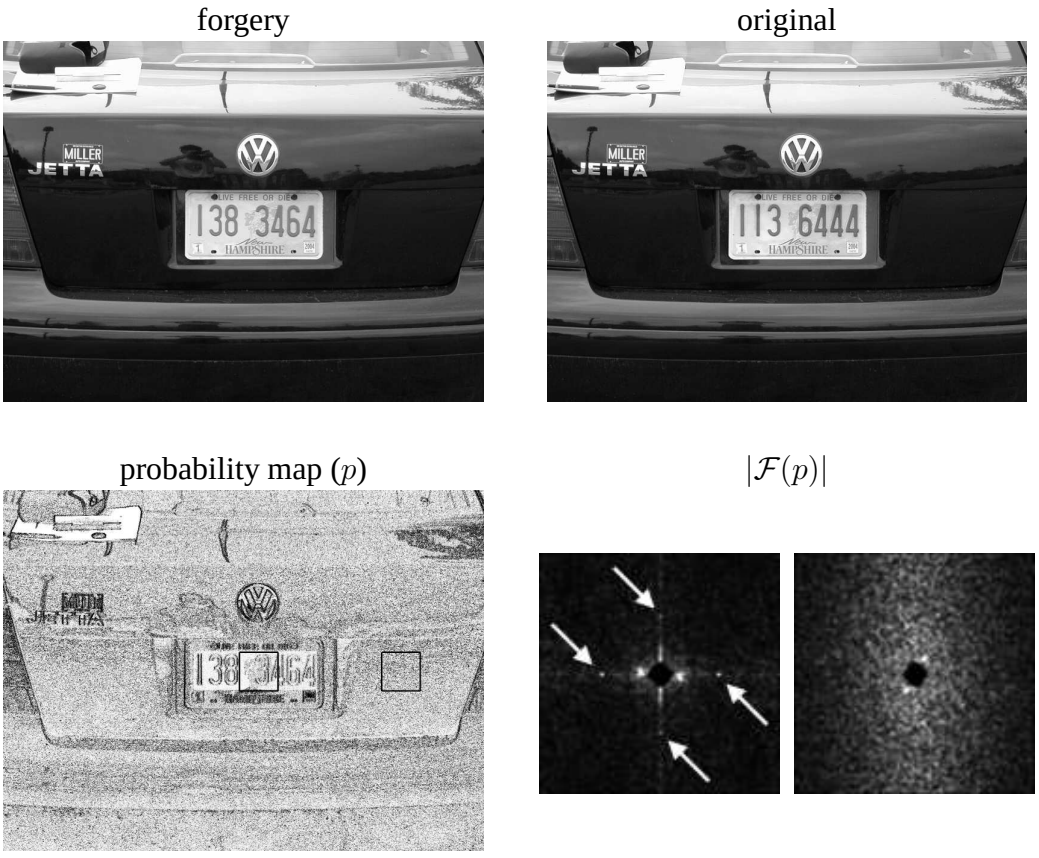

Figure 2.9: Shown are a forgery and the original image. The forgery consists of splicing in a new license plate number. Shown below is the estimated probability map (p) of the forgery, and the magnitude of the Fourier transform of regions in the license plate (left) and on the car trunk (right). The periodic pattern (spikes in $\mathcal{F}(p)$) in the license plate suggests that this region was re-sampled.

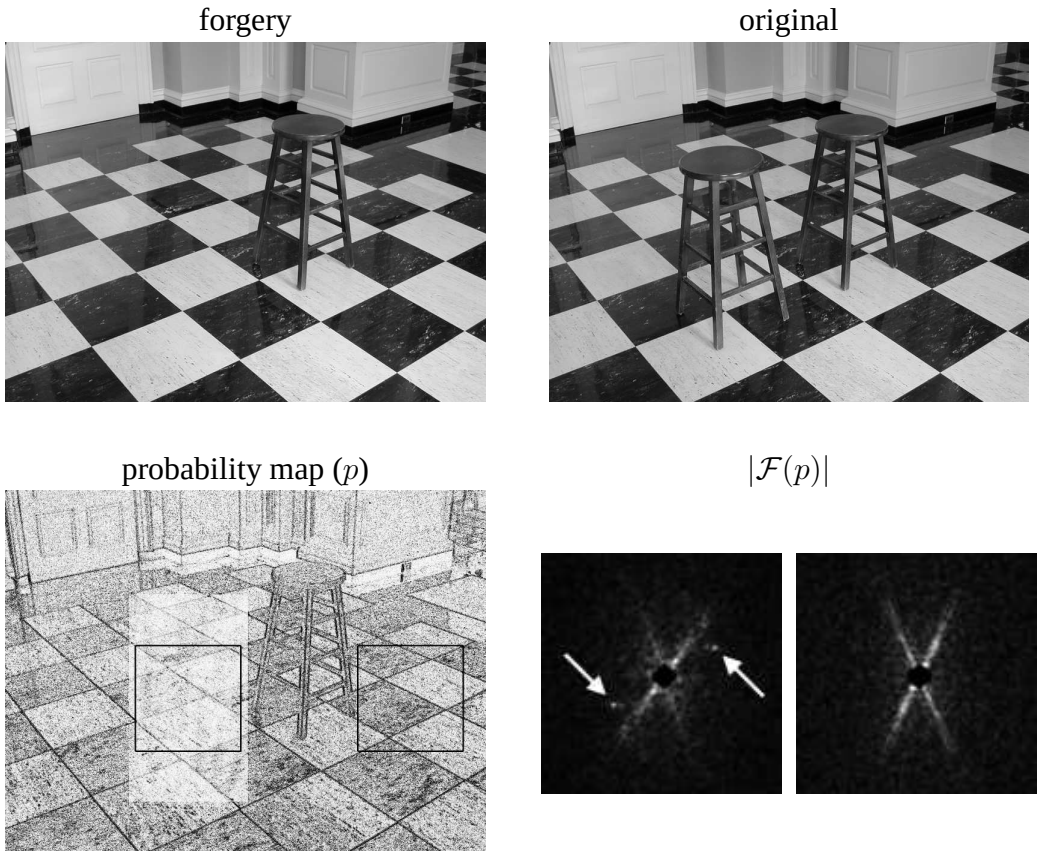

Figure 2.10: Shown are a forgery and the original image. The forgery consists of removing a stool by splicing in a new floor taken from another image (not shown here) of the same room. Shown below is the estimated probability map (p) of the forgery, and the magnitude of the Fourier transform of regions in the new floor (left) and in the original floor (right). The periodic pattern (spikes in $\mathcal{F}(p)$) in the new floor suggests that this region was re-sampled.

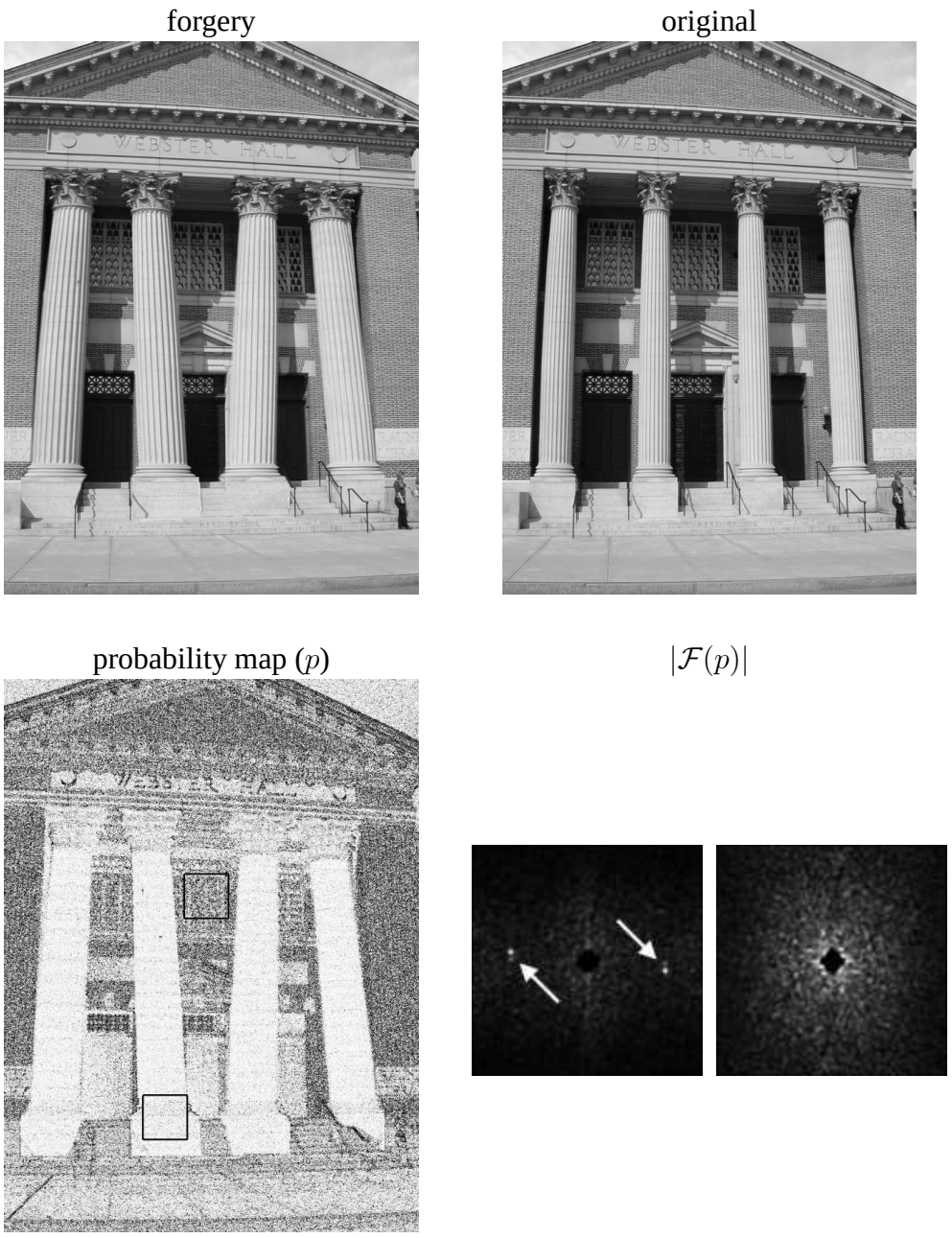

Figure 2.11: Shown are a forgery and the original image. The forgery consists of widening and shearing the pillars. Shown below is the estimated probability map (p) of the forgery, and the magnitude of the Fourier transform of regions on the pillar (left) and on the building (right). The periodic pattern (spikes in $\mathcal{F}(p)$) on the pillar suggests that this region was re-sampled.

and therefore we employ a simpler method that was experimentally determined to generate similar periodic patterns. This method first warps a rectilinear integer lattice according to a specified set of re-sampling parameters. From this warped lattice, the synthetic map is generated by computing the minimum distance between a warped point and an integer sampling lattice. More specifically, let M denote a general affine transform which embodies a specific re-sampling. Let (x, y) denote the points on an integer lattice, and $(\tilde{x}, \tilde{y})$ denote the points of a lattice obtained by warping the integer lattice (x, y) according to M:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} = M \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2.20}$$

The synthetic map, $s(x, y)$, corresponding to M is generated by computing the minimum distance between each point in the warped lattice $(\tilde{x}, \tilde{y})$ to a point in the integer lattice:

$$s(x, y) = \min_{x_0, y_0} \sqrt{(\tilde{x} - x_0)^2 + (\tilde{y} - y_0)^2}, \tag{2.21}$$

where x_0 and y_0 are integers, and $(\tilde{x}, \tilde{y})$ are functions of (x, y) as given in Equation (2.20). Synthetic maps generated using this method are similar to the experimentally determined probability maps, Figure 2.12.

The similarity between an estimated probability map, $p(x, y)$, and a synthetic map, $s(x, y)$, is computed as follows:

1. The probability map p is Fourier transformed: $P(\omega_x, \omega_y) = \mathcal{F}(p(x, y) \cdot W(x, y))$, where the radial portion of the rotationally invariant window, $W(x, y)$, takes the form:

$$f(r) = \begin{cases} 1 & 0 \le r < 3/4 \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi(r-3/4)}{\sqrt{2}-3/4}\right) & 3/4 \le r \le \sqrt{2}, \end{cases} \tag{2.22}$$

   where the radial axis is normalized between 0 and $\sqrt{2}$. For notational convenience the spatial arguments on $p(\cdot)$ and $P(\cdot)$ will be dropped.

2. The Fourier transformed map P is then high-pass filtered to remove undesired low frequency noise: $P_H = P \cdot H$, where the radial portion of the rotationally invariant high-pass filter, H, takes the form:

$$h(r) = \frac{1}{2} - \frac{1}{2}\cos\left(\frac{\pi r}{\sqrt{2}}\right), \quad 0 \le r \le \sqrt{2}. \tag{2.23}$$

3. The high-passed spectrum P_H is then normalized in the $[0, 1]$ range, gamma corrected to enhance frequency peaks, and then rescaled back to its original range:

$$P_G = \left(\frac{P_H}{\max(|P_H|)}\right)^4 \times \max(|P_H|). \tag{2.24}$$

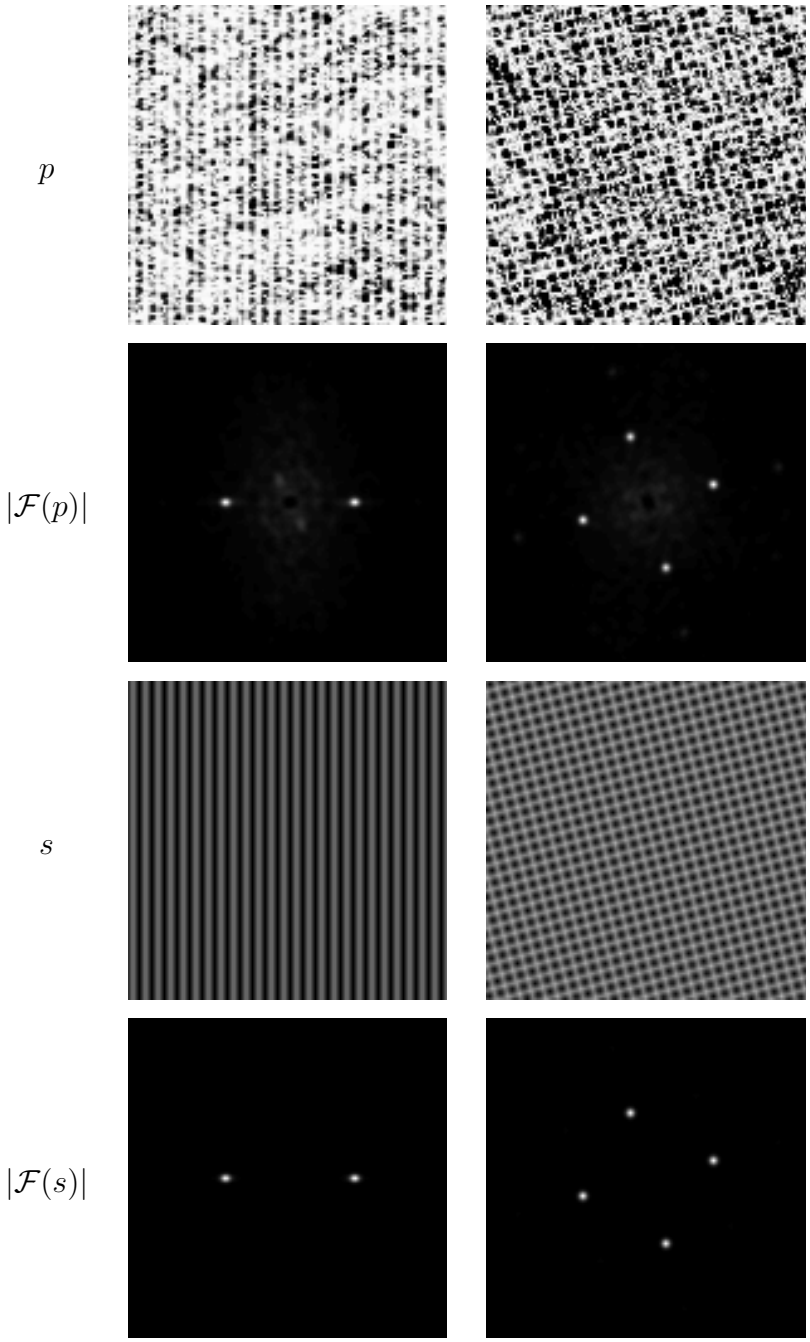

Figure 2.12: Shown in the first two rows are estimated probability maps, p, from images that were re-sampled (affine transformed), and the magnitude of the Fourier transform of these maps. Note the strong periodic patterns. Shown in the third and fourth rows are the synthetically generated probability maps computed using the same re-sampling parameters — note the similarity to the estimated maps.

4. The synthetic map s is simply Fourier transformed: $S = \mathcal{F}(s)$.

5. The measure of similarity between p and s is then given by:

$$M(p, s) = \sum_{\omega_x, \omega_y} |P_G(\omega_x, \omega_y)| \cdot |S(\omega_x, \omega_y)|, \tag{2.25}$$

where $|\cdot|$ denotes absolute value (note that this similarity measure is phase insensitive).

A set of synthetic probability maps are first generated from a number of different re-sampling parameters. For a given probability map p, the most similar synthetic map, $s^\star$, is found through a brute-force search over the entire set: $s^\star = \arg\max_s M(p, s)$. If the similarity measure, $M(p, s^\star)$, is above a specified threshold, then a periodic pattern is assumed to be present in the estimated probability map, and the image is classified as re-sampled. This threshold is empirically determined using only the original images in the database to yield a false positive rate less than 1%.

With the ability to quantitatively measure whether an image has been re-sampled, we tested the efficacy of our technique to detecting a range of re-sampling parameters, and the sensitivity to simple counter-measures that may be used to hide traces of re-sampling. In this analysis we employed the same set of images as described in the beginning of Section 2.4, and used the same set of algorithmic parameters. The images were re-sampled using bi-cubic interpolation. The probability map for a re-sampled image was estimated and compared against a large set of synthetic maps. For up-sampling, 160 synthetic maps were generated with re-sampling rates between 1% and 100%, in steps of 0.6%. For down-sampling, 160 synthetic maps were generated with re-sampling rates between 1% and 50%, in steps of 0.3%. For rotations, 45 synthetic maps were generated with rotation angles between 1° and 45°, in steps of 1°.

Uncompressed Images

Shown in Figure 2.13 are three graphs showing the detection accuracy for a range of up-sampling, down-sampling, and rotation rates. Each data point corresponds to the average detection accuracy over 50 gray-scale images. In these results, the false-positive rate (the probability of an original image to be incorrectly classified as re-sampled) is less than 1%. Note that detection is nearly perfect for up-sampling rates greater than 1%, and for rotations greater than 1°. As expected, the detection accuracy decreases as the down-sampling rate approaches 50%, Equation (2.16).

We have also measured the detection accuracy on affine transformed images. For simplicity we have employed affine transforms that consist in scaling followed by rotation, i.e., if M denotes the affine transform matrix, then $M = RS$, where S is a diagonal matrix (scaling) and R is an orthogonal matrix (rotation). Results are shown in Table 2.1. Shown on the rows are the detection accuracies obtained when the rotation angle is fixed and the scaling parameter[6] varies, while on the columns are the detection accuracies when the scaling parameter is held constant and the rotation angle varies. Note that the detection accuracy is typically governed by the smallest detection

[6]Note that a scaling parameter less than one corresponds to up-sampling, while a scaling parameter more than one corresponds to down-sampling.

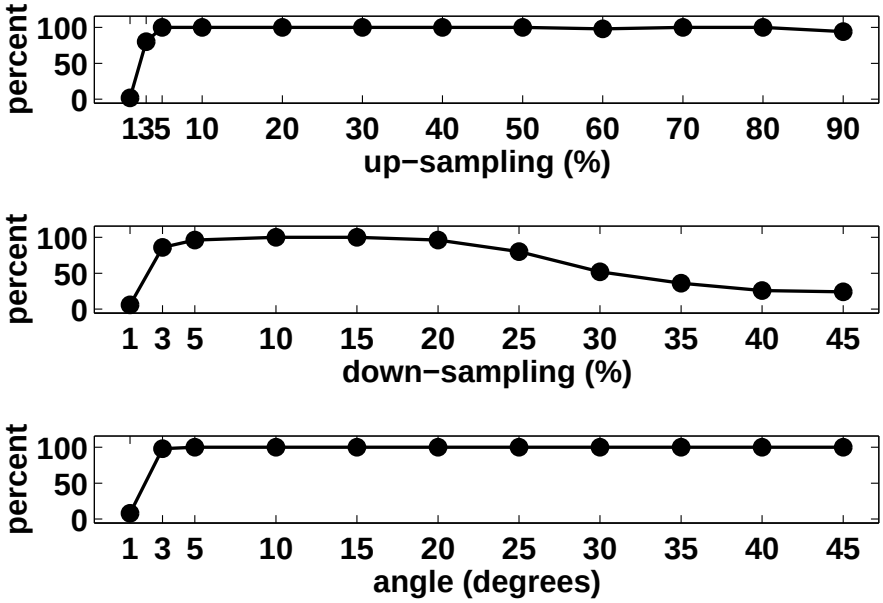

Figure 2.13: Detection accuracy as a function of different re-sampling parameters. Each data point corresponds to the average detection accuracy over 50 images.

-	0.60	0.70	0.80	0.90	1.05	1.10	1.15	1.20	1.25	1.30	1.35	1.40
$1°$	100%	100%	100%	100%	88%	100%	96%	100%	88%	52%	80%	92%
$2°$	100%	100%	100%	100%	96%	96%	100%	100%	76%	52%	68%	92%
$3°$	100%	100%	100%	100%	100%	100%	96%	100%	64%	68%	72%	92%
$4°$	100%	100%	100%	100%	100%	96%	100%	88%	92%	32%	56%	92%
$5°$	100%	100%	100%	100%	100%	96%	96%	96%	80%	64%	60%	92%
$10°$	100%	100%	100%	100%	100%	100%	100%	96%	80%	24%	32%	92%
$15°$	100%	100%	100%	100%	100%	100%	100%	96%	44%	20%	32%	92%
$20°$	100%	100%	100%	100%	100%	100%	100%	92%	48%	40%	24%	92%
$25°$	100%	100%	100%	100%	100%	100%	100%	96%	80%	60%	32%	92%
$30°$	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	92%
$35°$	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	92%
$40°$	100%	100%	100%	100%	100%	100%	100%	100%	96%	100%	100%	92%
$45°$	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	92%	92%

Table 2.1: Accuracies for affi ne transforms consisting of a scaling followed by a rotation. Each accuracy estimate represents an average over 50 images.

accuracy of the multiple re-samplings, i.e. for small rotation angles and for large down-sampling factors the detection accuracy decreases.

The generalization of our algorithm to color images is fairly straightforward — each color channel is independently subjected to the same analysis as that employed for gray-scale images. Shown in Figures 2.14-2.15 are the detection accuracies for each color channel treated as an independent gray-scale image, as well as the accuracy when at least one channel is detected as re-sampled. These accuracies were estimated over the same ranges of up-sampling, down-sampling, and rotation rates as those used in the gray-scale experiments. Each data point corresponds to an average detection accuracy over 50 color images, with a false-positive rate of less than 1%. Note that these results are very similar to those obtained for gray-scale images, with a slight improvement for larger down-sampling rates.

Robustness to Simple Counter-attacks

In a forensic context, the robustness of our detection algorithm is of particular interest. To this end, we have tested our method's capability to resist two simple counter-attacks: additive white Gaussian noise and non-linear gamma correction. Specifically, after re-sampling an image we either added a controlled amount of white Gaussian noise, or applied gamma correction with varying gamma values.

Shown in Figures 2.16-2.17 are the detection accuracies for a range of up-sampling, down-sampling, and rotation rates, and for a range of reasonable gamma values. Note that the detection accuracy is practically unaffected by non-linear gamma correction for a large range of gamma values. Shown in Figures 2.18-2.19 are the detection accuracies for the same range of up-sampling, down-sampling, and rotation rates, and for a range of reasonably chosen signal-to-noise ratios (SNR). Note that the performance for up-sampling and rotation remains good even for low SNRs, e.g., 26 db, except for lower up-sampling percentages ($1\% - 5\%$) and smaller rotation angles ($1^\circ - 5^\circ$). In the case of down-sampling the drop in the detection accuracy is more pronounced, and can be particularly noticed between 35 db and 31 db. Each data point corresponds to the average detection accuracy over 50 images, with a false-positive rate of less than 1%, in both the white noise and gamma correction experiments.

It may seem surprising that, although the additive noise or non-linear luminance transforms may destroy or alter the correlations introduced by re-sampling, our algorithm can still detect these correlations. The reason for this robustness is that the EM algorithm employs a model of correlations that are already corrupted by noise, Equation (2.10). As a result, the accuracy of re-sampling detection is largely insensitive to small perturbations, and drops gracefully when the amount of perturbation increases.

Robustness to JPEG, JPEG2000, and GIF

We have also tested our algorithm against images compressed with lossy compression algorithms: JPEG [1], GIF, and JPEG2000 [3]. Specifically, gray-scale (for JPEG and JPEG 2000) and color (for GIF) images were subjected to re-sampling with a range of parameters, then compressed and

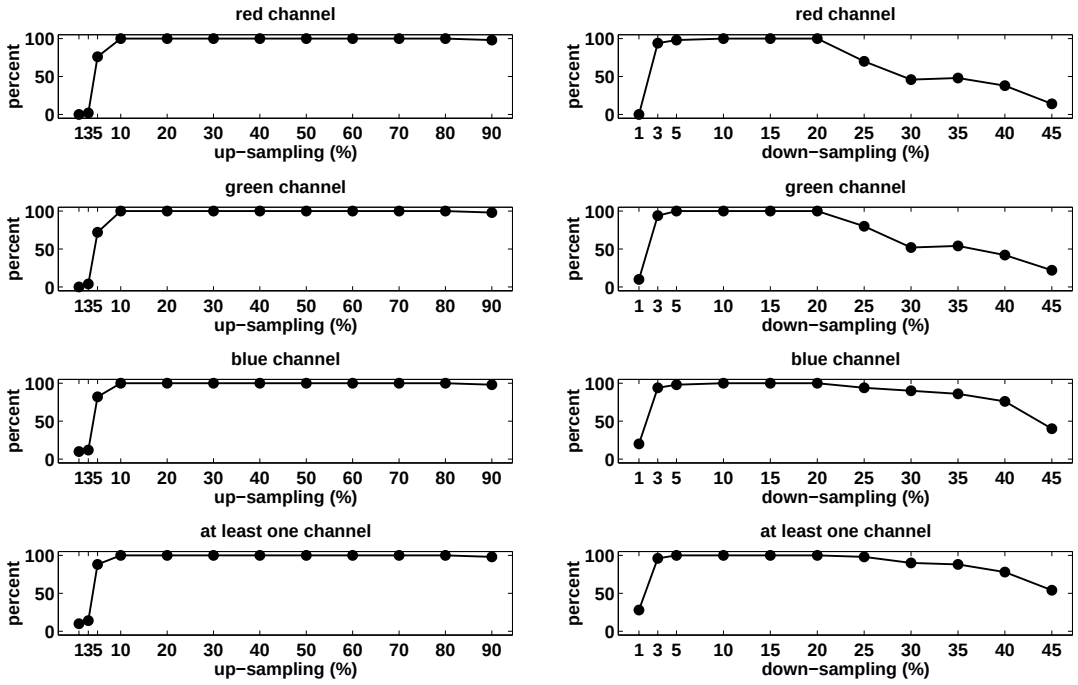

Figure 2.14: Detection accuracy as a function of different up- and down-sampling percentages for different channels of color images. Each data point corresponds to the average detection accuracy over 50 images.

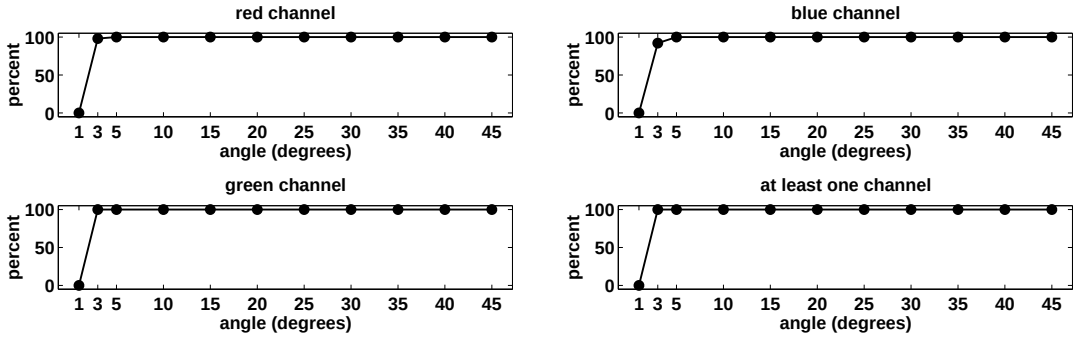

Figure 2.15: Detection accuracy as a function of different rotation angles for different channels of color images. Each data point corresponds to the average detection accuracy over 50 images.

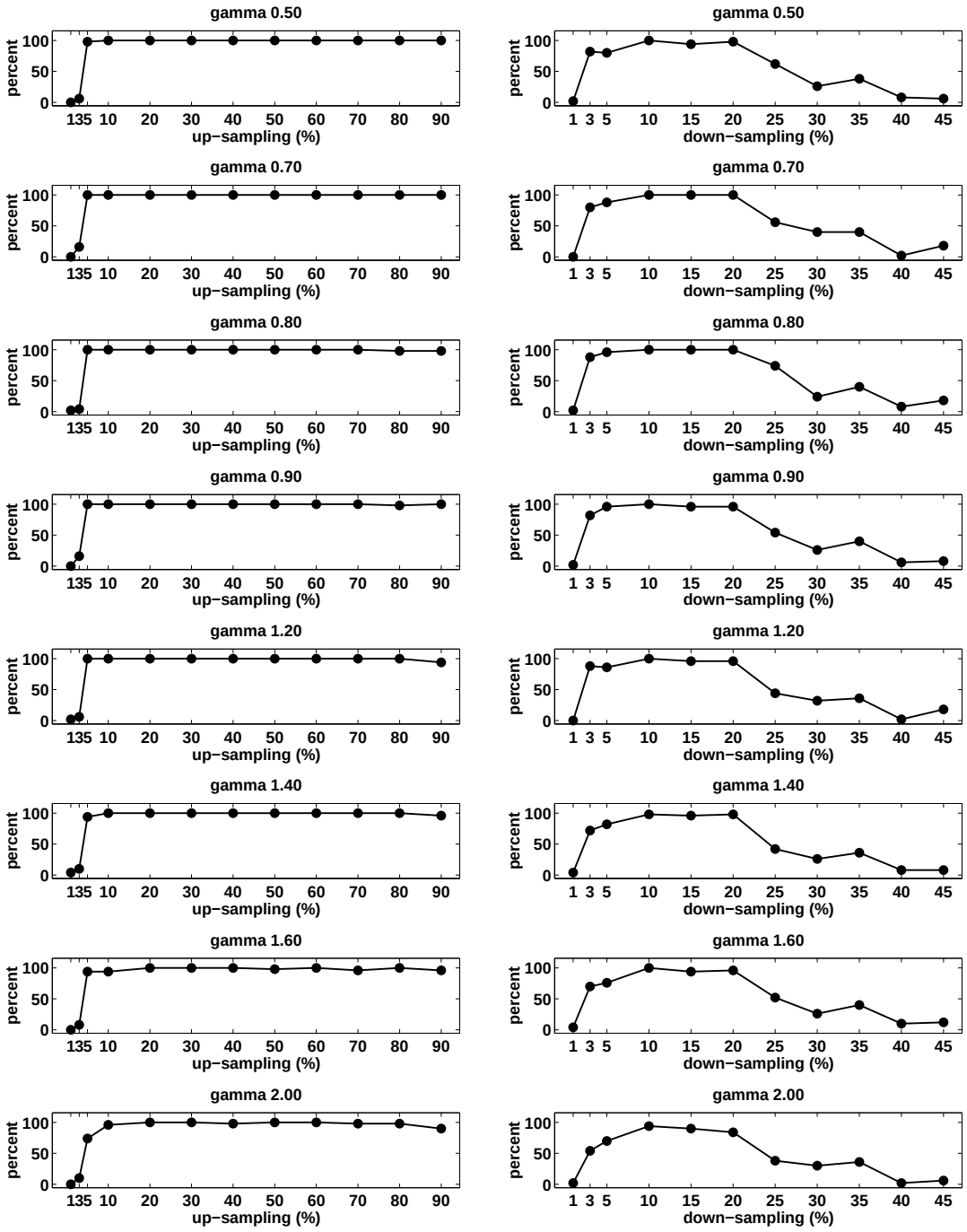

Figure 2.16: Detection accuracy as a function of different up- and down-sampling percentages and gamma values. Each data point corresponds to the average detection accuracy over 50 images.

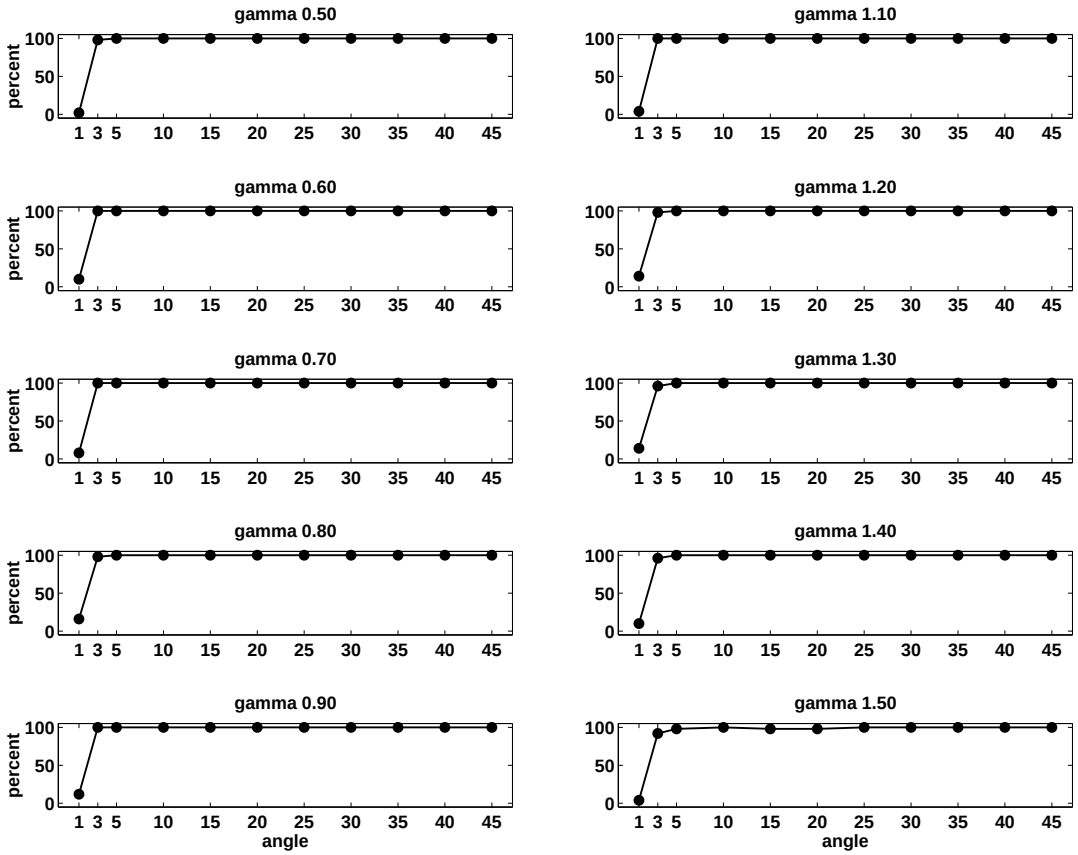

Figure 2.17: Detection accuracy as a function of different rotation angles and gamma values. Each data point corresponds to the average detection accuracy over 50 images.

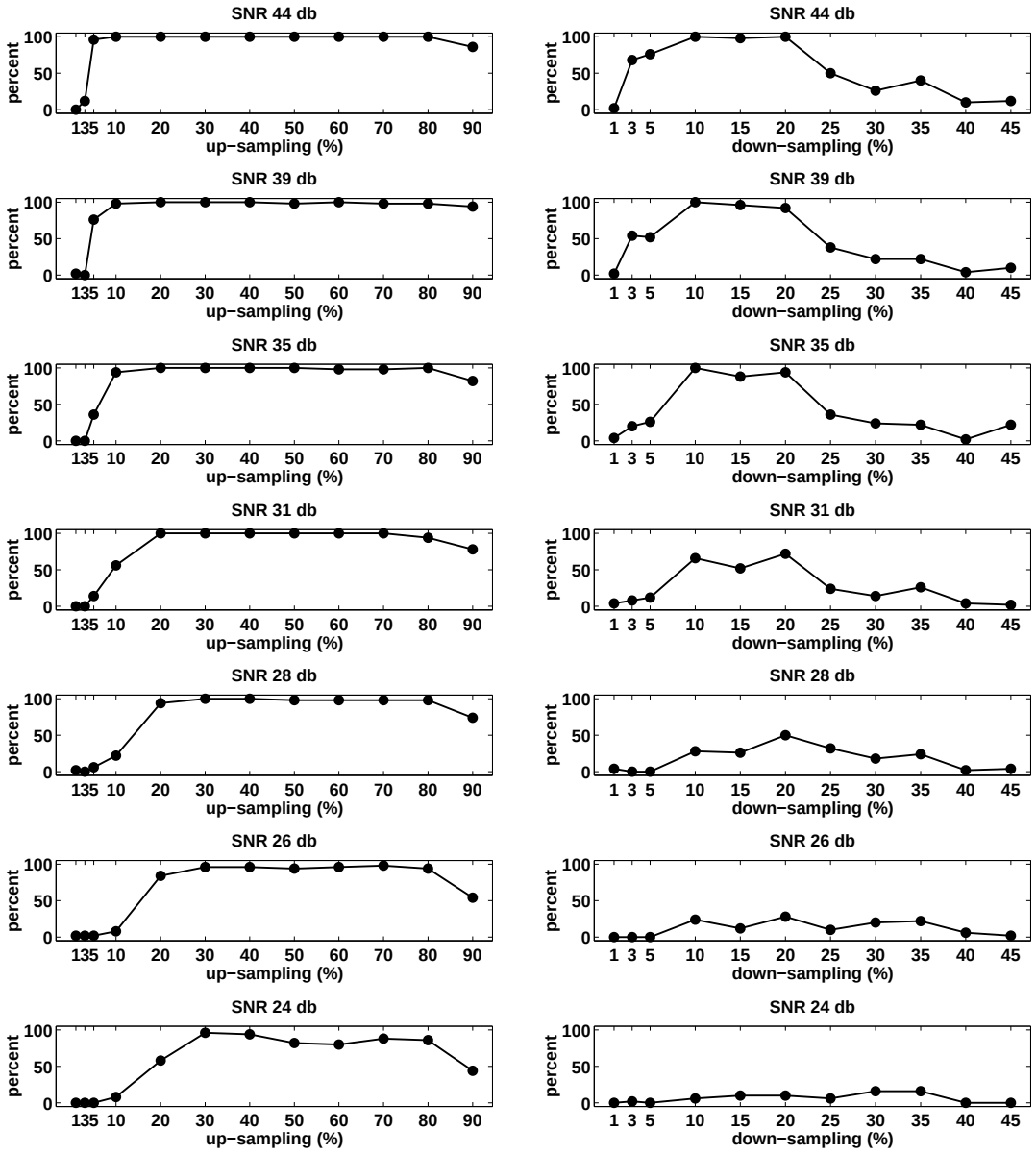

Figure 2.18: Detection accuracy as a function of different up- and down-sampling percentages and SNRs. Each data point corresponds to the average detection accuracy over 50 images.

stored in JPEG, GIF, or JPEG2000 formats. For all the experiments presented below, each data point corresponds to an average over 50 images, and the false-positive rate is less than 1%.

Shown in Figures 2.20-2.22 are the detection accuracies for re-sampled gray-scale images that were subsequently JPEG compressed with different qualities. These results reveal a weakness of our method: while the detection accuracies are good at qualities between 97 and 100 (out of 100) there is a precipitous fall at quality 96 for down-samplings and rotations, and at a quality of 90, the detection is nearly at chance for all re-sampling rates. Note also that at an up-sampling rate of 60% and a down-sampling rate of 20% the detection accuracy drops suddenly. This is because the periodic JPEG blocking artifacts happen to coincide with the periodic patterns introduced by these re-sampling parameters — these artifacts do not interfere with the detection of rotations. The reason for the general poor performance of detecting re-sampling in JPEG compressed images is two-fold. First, lossy JPEG compression introduces correlated non-white noise into the image (e.g., a compression quality of 90 introduces, on average, 28 db of noise), which significantly affects the detection accuracy. Second, and more important, the block artifacts introduced by JPEG introduce very strong periodic patterns that mask and interfere with the periodic patterns introduced by re-sampling.

Shown in Figures 2.23-2.24 are the detection accuracies for re-sampled color images that were subsequently converted to 8-bit indexed color format (GIF). This conversion introduces approximately 25 db of noise. Note that while not as good as the uncompressed TIFF images, these detection rates are slightly superior (for down-sampling in particular) to what would be expected with the level of noise (see Figures 2.18-2.19) introduced by GIF compression.

We have also tested our method's robustness with JPEG2000 [3] compression. We have employed a free, non-official Java™ implementation of the JPEG2000 baseline codec, JJ2000 [54], freely available to download at

`http://jj2000.epfl.ch/jj_download/index.html`.

Shown in Figures 2.26-2.25 are the results for gray-scale images that were first re-sampled with a range of parameters, then compressed with different numbers of bits per pixel using the JPEG2000 codec. Note that the detection remains robust down to 2 bits per pixel. A significant deterioration is experienced for compression rates below 1.5 bits per pixel (for down-sampling in particular). Note also that the detection accuracy for higher up-sampling rates is lower when compared to TIFF images. The reason for this decrease in accuracy is that JPEG2000 introduces artifacts in the probability maps that interfere with periodic high frequency patterns like those produced by up-samplings with higher percentages. These artifacts, however, are not as severe as those introduced by JPEG, hence the improved overall performance.

2.5 Summary

When creating digital forgeries, it is often necessary to scale, rotate, or distort a portion of an image. This process involves re-sampling the original image onto a new lattice. Although this re-sampling process typically leaves behind no perceptual artifacts, it does introduce periodic correlations between neighboring image pixels. We have shown how and when correlation patterns

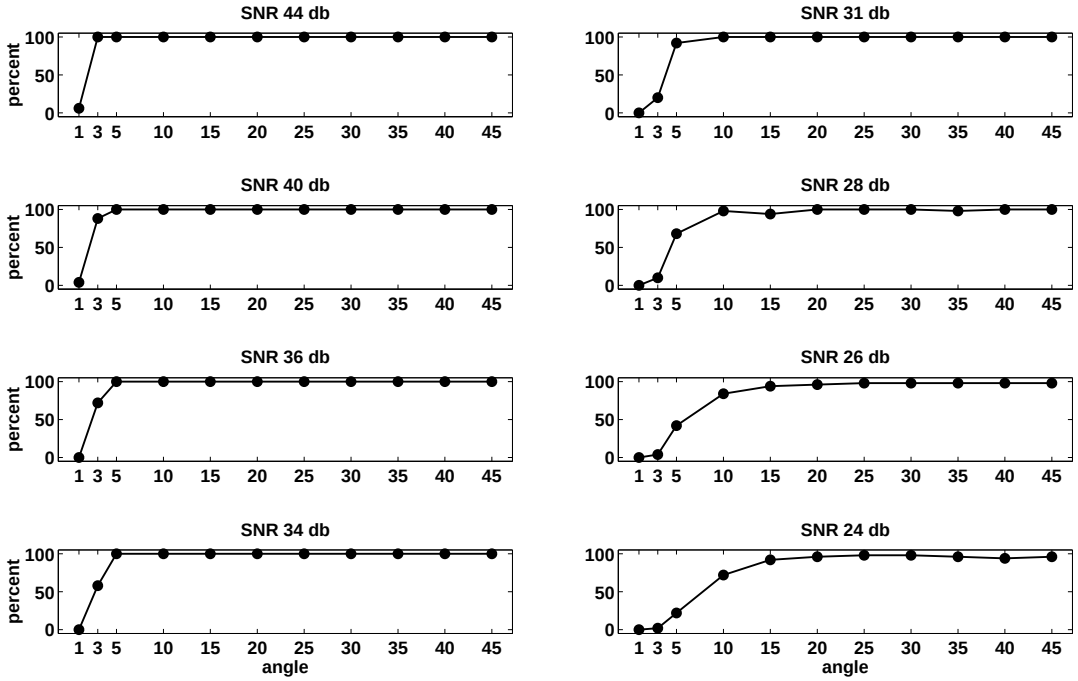

Figure 2.19: Detection accuracy as a function of different rotation angles and SNRs. Each data point corresponds to the average detection accuracy over 50 images.

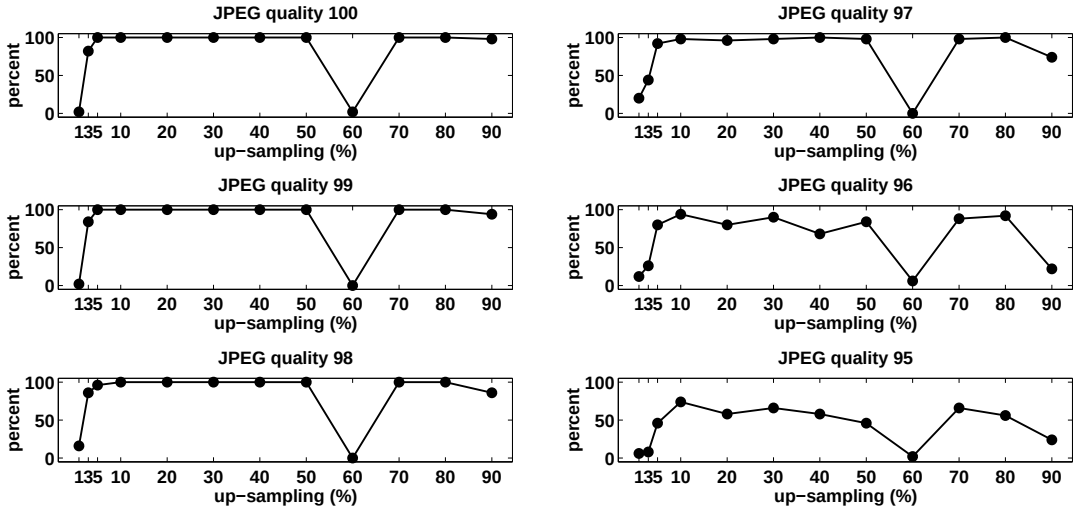

Figure 2.20: Detection accuracy as a function of different up-sampling percentages and JPEG compression qualities. Each data point corresponds to the average detection accuracy over 50 images.

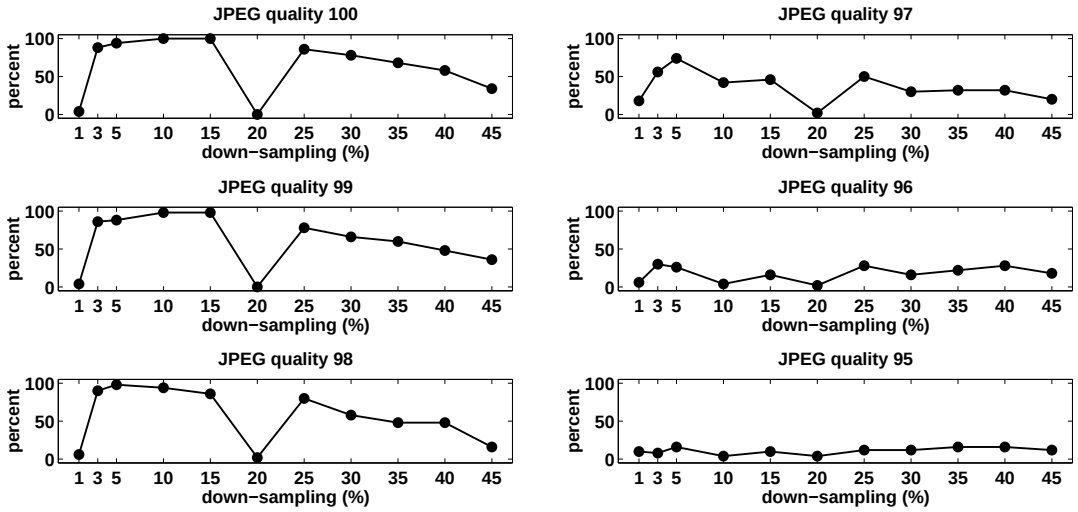

Figure 2.21: Detection accuracy as a function of different down-sampling percentages and JPEG compression qualities. Each data point corresponds to the average detection accuracy over 50 images.

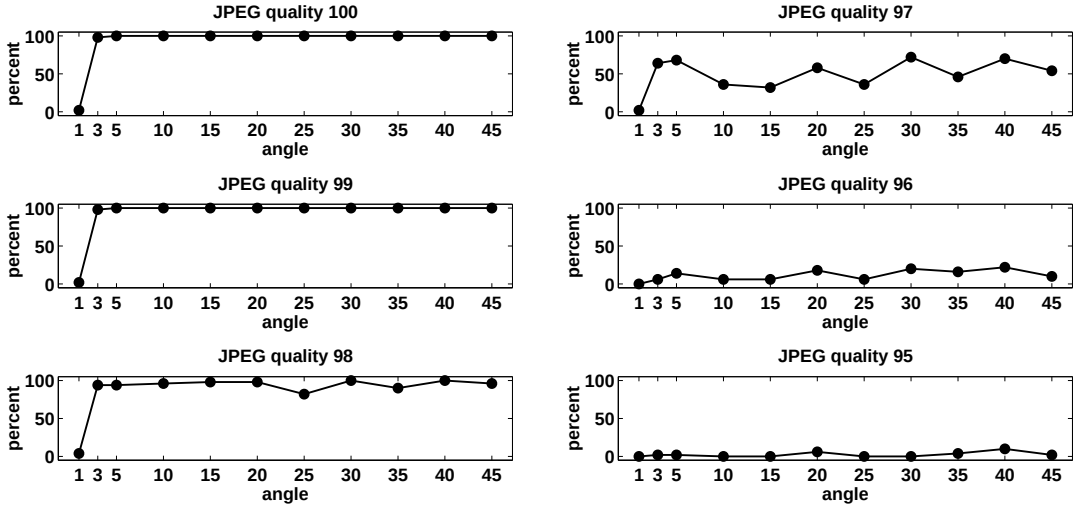

Figure 2.22: Detection accuracy as a function of different rotation angles and JPEG compression qualities. Each data point corresponds to the average detection accuracy over 50 images.

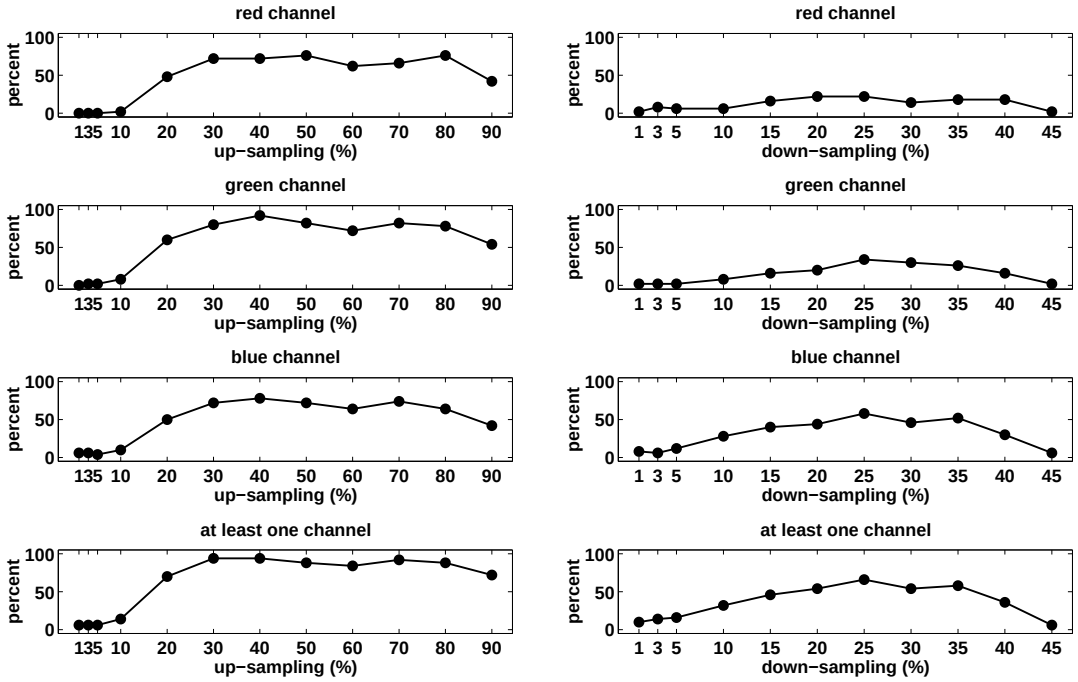

Figure 2.23: Detection accuracy as a function of different up- and down-sampling percentages for different channels of color images stored in GIF format. Each data point corresponds to the average detection accuracy over 50 images.

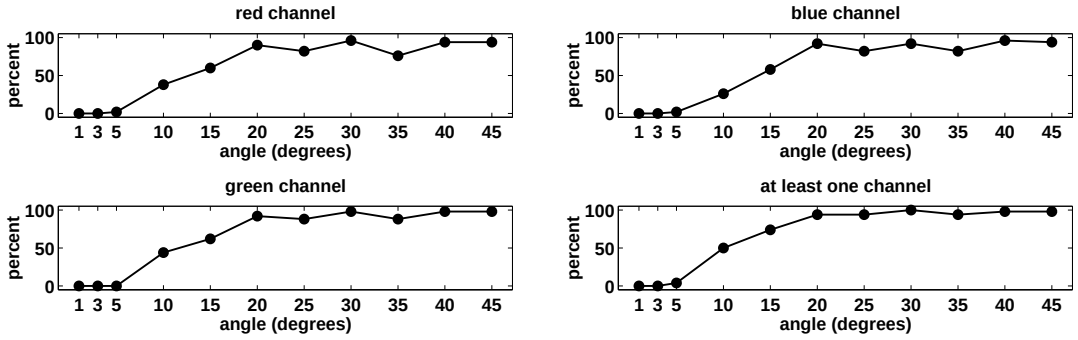

Figure 2.24: Detection accuracy as a function of different rotation angles for different channels of color images stored in GIF format. Each data point corresponds to the average detection accuracy over 50 images.

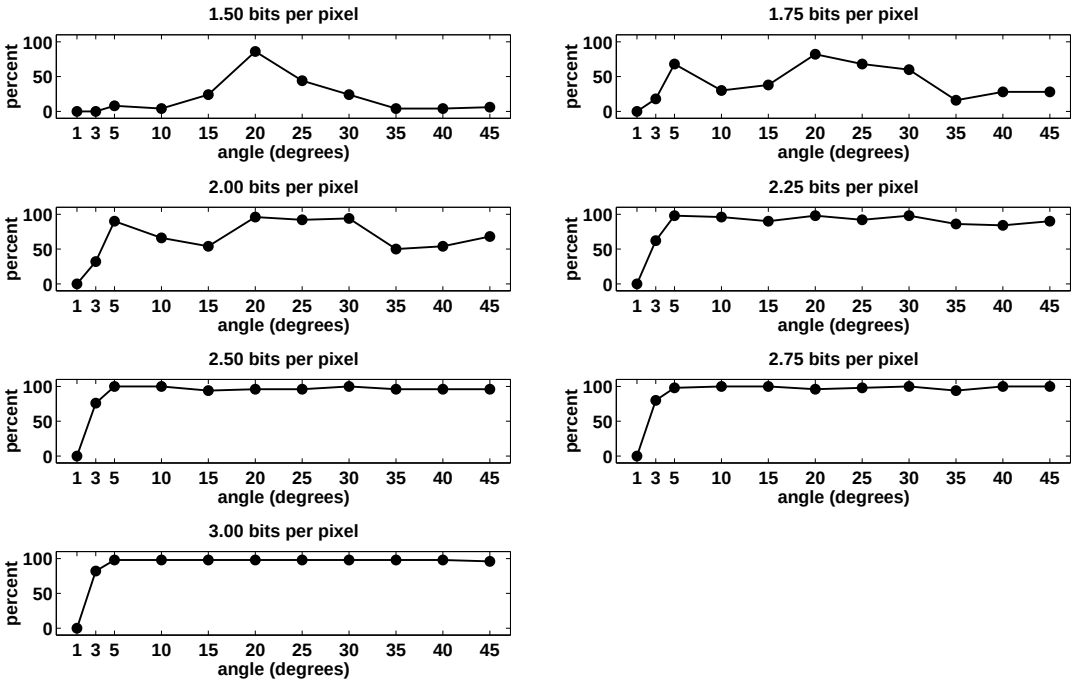

Figure 2.25: Detection accuracy as a function of different rotation angles and JPEG2000 compression in bits per pixel. Each data point corresponds to the average detection accuracy over 50 images.

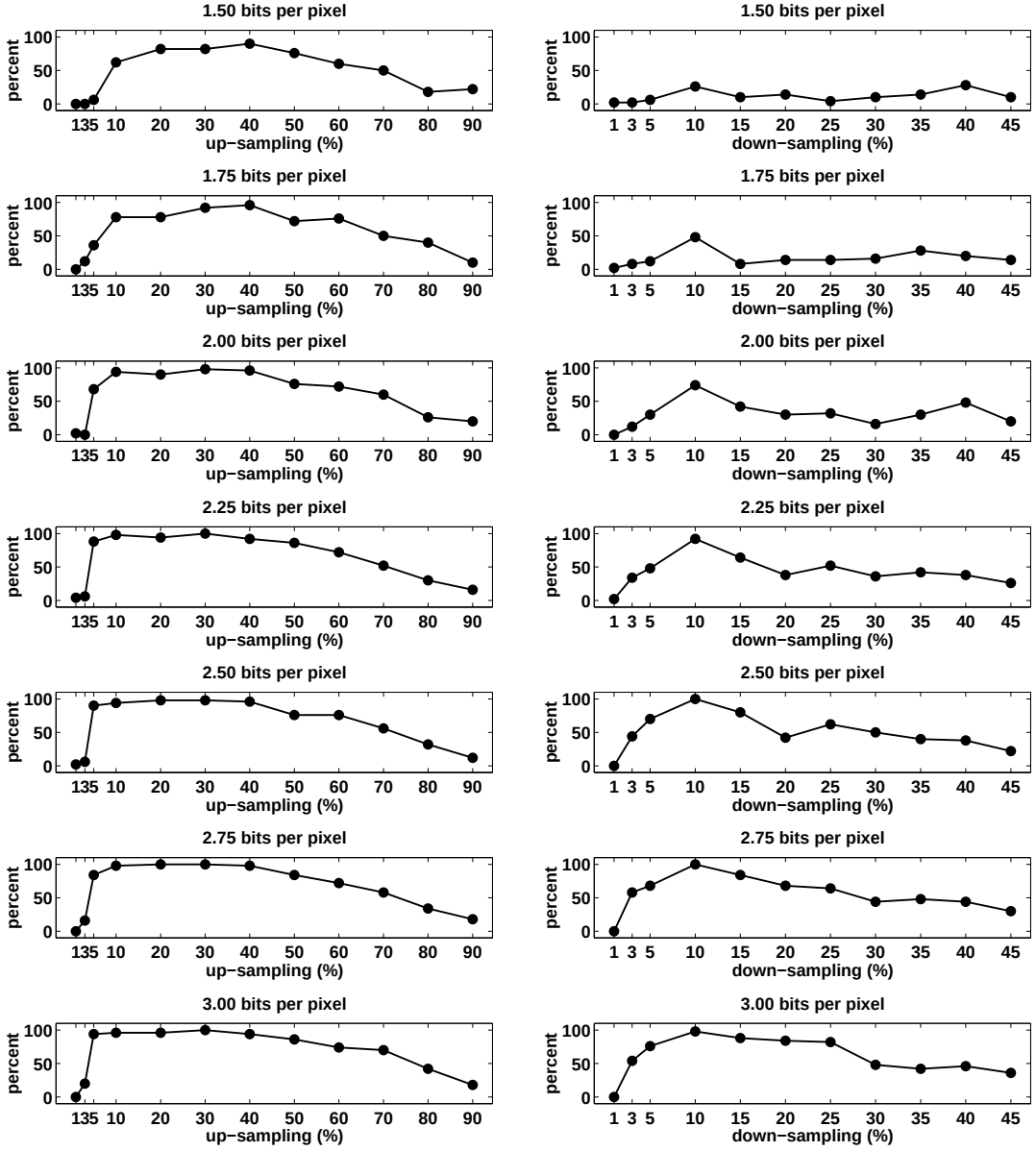

Figure 2.26: Detection accuracy as a function of different up-sampling (left) and down-sampling (right) percentages and JPEG2000 compression in bits per pixel. Each data point corresponds to the average detection accuracy over 50 images.

are introduced, and described a technique to detect them in any portion of an image. We have shown several detection results on natural test images and credible forgeries, and we have thoroughly tested our method's robustness and sensitivity to simple counter-attacks (additive white Gaussian noise and non-linear luminance transformations) and lossy compression schemes (GIF, JPEG2000, and JPEG). In summary, we have shown that for uncompressed TIFF images, and lossy compressed images with minimal compression we can detect whether an image, or a portion of it, has been re-sampled (scaled, rotated, etc.), as might occur when the image has been tampered with.

2.6 Re-sampling Detection Algorithm

/* Initialize */

choose a random $\vec{\alpha}_0$

choose N and σ_0

set p_0 as 1 over the size of the range of possible values for $y(i)$

set Y as in Equation (2.15)

set h to be a binomial low-pass filter of size $(N_h \times N_h)$

$n = 0$

repeat

/* expectation step */

for each sample i

$R(i) = \left| y(i) - \sum_{k=-N}^{N} \alpha_n(k) y(i+k) \right|$ /* residual error */

end

$R = R \star h$ /* spatially average the residual error */

for each sample i

$P(i) = \frac{1}{\sigma_n \sqrt{2\pi}} e^{-R^2(i)/2\sigma_n^2}$ /* conditional probability */

$w(i) = \frac{P(i)}{P(i)+p_0}$ /* posterior probability */

end

/* maximization step */

W = 0

for each sample i

$W(i,i) = w(i)$ /* weighting matrix */

end

$\sigma_{n+1} = \left(\frac{\sum_i w(i)\, R^2(i)}{\sum_i w(i)} \right)^{1/2}$ /* new variance estimate */

$\vec{\alpha}_{n+1} = (Y^T W Y)^{-1} Y^T W \vec{y}$ /* new estimate */

$n = n + 1$

until ($\|\vec{\alpha}_n - \vec{\alpha}_{n-1}\| < \epsilon$) /* stopping condition */

Chapter 3

Color Filter Array Interpolated Images

Most digital cameras capture color images using a single sensor in conjunction with an array of color filters. As a result, only one third of the samples in a color image are captured by the camera, the other two thirds being interpolated. This interpolation introduces specific correlations between the samples of a color image. When creating digital forgeries these correlations are likely to be destroyed or altered. We describe the form of these correlations, and propose a method that quantifies and estimates them in any portion of an image. We show the general effectiveness of this technique in detecting tampered images, and analyze its sensitivity and robustness to simple counter-attacks.

3.1 Color Filter Array Interpolation Algorithms

A digital color image consists of three channels containing samples from different bands of the color spectrum (e.g., red, green, and blue). Most digital cameras, however, are equipped with a single charge-coupled device (CCD) or complementary metal oxide semiconductor (CMOS) sensor, and capture color images using a color filter array (CFA). The most frequently used color filter array (CFA), the Bayer array [4], employs three color filters: red, green, and blue. The red and blue pixels are sampled on rectilinear lattices, while the green pixels are sampled on a quincunx lattice, Figure 3.1. Since only a single color sample is recorded at each pixel location, the other two color samples must be estimated from the neighboring samples in order to obtain a three-channel color image. Let $S(x,y)$ denote the CFA image in Figure 3.1, and $\tilde{R}(x,y)$, $\tilde{G}(x,y)$, $\tilde{B}(x,y)$ denote the red, green, and blue channels constructed from $S(x,y)$ by inserting zeros at the locations of missing color samples:

$$\tilde{R}(x,y) = \begin{cases} S(x,y) & \text{if } S(x,y) = r_{x,y} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$$\tilde{G}(x,y) = \begin{cases} S(x,y) & \text{if } S(x,y) = g_{x,y} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

$r_{1,1}$	$g_{1,2}$	$r_{1,3}$	$g_{1,4}$	$r_{1,5}$	$g_{1,6}$	
$g_{2,1}$	$b_{2,2}$	$g_{2,3}$	$b_{2,4}$	$g_{2,5}$	$b_{2,6}$	
$r_{3,1}$	$g_{3,2}$	$r_{3,3}$	$g_{3,4}$	$r_{3,5}$	$g_{3,6}$	
$g_{4,1}$	$b_{4,2}$	$g_{4,3}$	$b_{4,4}$	$g_{4,5}$	$b_{4,6}$	$\cdots$
$r_{5,1}$	$g_{5,2}$	$r_{5,3}$	$g_{5,4}$	$r_{5,5}$	$g_{5,6}$	
$g_{6,1}$	$b_{6,2}$	$g_{6,3}$	$b_{6,4}$	$g_{6,5}$	$b_{6,6}$	
			$\vdots$			$\ddots$

Figure 3.1: The top-left portion of a CFA image obtained from a Bayer array. The red pixels, $r_{2i+1,2j+1}$, and blue pixels, $b_{2i,2j}$, are sampled on rectilinear lattices, while the green pixels, $g_{2i+1,2j}$ and $g_{2i,2j+1}$, are sampled twice as often on a quincunx lattice. Only one color sample is recorded at each pixel location.

$$\tilde{B}(x,y) \;=\; \begin{cases} S(x,y) & \text{if } S(x,y) = b_{x,y} \\ 0 & \text{otherwise} \end{cases}, \tag{3.3}$$

where (x, y) span an integer lattice. A complete color image, with channels $R(x, y)$, $G(x, y)$, and $B(x, y)$ needs to be estimated. These channels take on the non-zero values of $\tilde{R}(x, y)$, $\tilde{G}(x, y)$, and $\tilde{B}(x, y)$, and replace the zeros with estimates from neighboring samples. The estimation of the missing color samples is referred to as CFA interpolation or demosaicking. CFA interpolation has been extensively studied and many methods have been proposed (see, for example, [55] for a survey, and [27, 29, 46] for more recent methods). We will briefly describe seven demosaicking techniques for the Bayer array that are used in our studies.

3.1.1 Bilinear and Bi-Cubic

The simplest methods for demosaicking are kernel-based interpolation methods that act independently on each channel. These methods can be efficiently implemented as linear filtering operations on each color channel:

$$\begin{aligned} R(x,y) &= \sum_{u,v=-N}^{N} h_r(u,v)\tilde{R}(x-u,y-v) \\ G(x,y) &= \sum_{u,v=-N}^{N} h_g(u,v)\tilde{G}(x-u,y-v) \\ B(x,y) &= \sum_{u,v=-N}^{N} h_b(u,v)\tilde{B}(x-u,y-v), \end{aligned} \tag{3.4}$$

where $\tilde{R}(\cdot)$, $\tilde{G}(\cdot)$, $\tilde{B}(\cdot)$ are defined in Equations (3.1)-(3.3), and $h_r(\cdot)$, $h_g(\cdot)$, $h_b(\cdot)$ are linear filters of size $(2N + 1) \times (2N + 1)$. Different forms of interpolation (nearest neighbor, bilinear, bi-

cubic [32], etc.) differ in the form of the interpolation filter used. For example, in the case of a Bayer array, the bilinear and bi-cubic filters for the red and blue channels are separable, with the 1-D filters given by:

$$h_l = \begin{bmatrix} 1/2 & 1 & 1/2 \end{bmatrix} \qquad h_c = \begin{bmatrix} -1/16 & 0 & 9/16 & 1 & 9/16 & 0 & -1/16 \end{bmatrix} . \tag{3.5}$$

and the 2-D filters given by $h_l^T h_l$ and $h_c^T h_c$. The bilinear and bi-cubic filters for the green channel, however, are no longer separable and take the form:

$$h_l = \frac{1}{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad h_c = \frac{1}{256}\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -9 & 0 & -9 & 0 & 0 \\ 0 & -9 & 0 & 81 & 0 & -9 & 0 \\ 1 & 0 & 81 & 256 & 81 & 0 & 1 \\ 0 & -9 & 0 & 81 & 0 & -9 & 0 \\ 0 & 0 & -9 & 0 & -9 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} . \tag{3.6}$$

Using bilinear interpolation, for example, the color samples at pixel locations $(3,3)$, $(3,4)$, and $(4,4)$, Figure 3.1, are given by:

$$\begin{aligned} R(3,3) &= r_{3,3} & G(3,3) &= \frac{g_{2,3}+g_{3,2}+g_{3,4}+g_{4,3}}{4} & B(3,3) &= \frac{b_{2,2}+b_{2,4}+b_{4,2}+b_{4,4}}{4} \\ G(3,4) &= g_{3,4} & B(3,4) &= \frac{b_{2,4}+b_{4,4}}{2} & R(3,4) &= \frac{r_{3,3}+r_{3,5}}{2} \\ B(4,4) &= b_{4,4} & R(4,4) &= \frac{r_{3,3}+r_{3,5}+r_{5,3}+r_{5,5}}{4} & G(4,4) &= \frac{g_{3,4}+g_{4,3}+g_{4,5}+g_{5,4}}{4} . \end{aligned}$$

3.1.2 Smooth Hue Transition

Smooth hue transition CFA interpolation [10] is based on the assumption that color hue varies smoothly in natural images, and can be assumed to be constant in a small neighborhood. Hue is defined here as the ratio between a chrominance component (red or blue) and the luminance component (green). The algorithm interpolates the green channel $\tilde{G}(\cdot)$, Equation (3.2), using bilinear interpolation, Section 3.1.1, to yield $G(\cdot)$. To estimate the missing red samples, the ratio $\tilde{R}(\cdot)/G(\cdot)$ is bilinearly interpolated, then point-wise multiplied by $G(\cdot)$. For example, the missing red values at locations $(3,4)$, $(4,3)$, and $(4,4)$, Figure 3.1, are given by:

$$\begin{aligned} R(3,4) &= G(3,4)\cdot\frac{1}{2}\left(\frac{r_{3,3}}{G(3,3)}+\frac{r_{3,5}}{G(3,5)}\right) \\ R(4,3) &= G(4,3)\cdot\frac{1}{2}\left(\frac{r_{3,3}}{G(3,3)}+\frac{r_{5,3}}{G(5,3)}\right) \\ R(4,4) &= G(4,4)\cdot\frac{1}{4}\left(\frac{r_{3,3}}{G(3,3)}+\frac{r_{3,5}}{G(3,5)}+\frac{r_{5,3}}{G(5,3)}+\frac{r_{5,5}}{G(5,5)}\right) . \end{aligned} \tag{3.7}$$

The missing blue samples are estimated in a similar manner as the red samples.

3.1.3 Median Filter

Median filter based CFA interpolation [20] is a two-step process. First the $\tilde{R}(\cdot)$, $\tilde{G}(\cdot)$, $\tilde{B}(\cdot)$ channels, Equations (3.1)-(3.3), are each interpolated using bilinear interpolation, Section 3.1.1. Next, the pairwise differences of the interpolated channels (i.e., red minus green, red minus blue, and green minus blue) are median filtered[1]. Let $M_{rg}(\cdot)$, $M_{gb}(\cdot)$, and $M_{rb}(\cdot)$ denote the median filtered differences. At each pixel location, the missing color samples are estimated as the sum or difference between the color samples from the CFA image, Figure 3.1, and the corresponding median filtered differences. For example, the missing samples at locations $(3,3)$, $(3,4)$, and $(4,4)$ are estimated as follows:

$$\begin{aligned} G(3,3) &= r_{3,3} - M_{rg}(3,3) & B(3,3) &= r_{3,3} - M_{rb}(3,3) \\ R(3,4) &= g_{3,4} + M_{rg}(3,4) & B(3,4) &= g_{3,4} - M_{gb}(3,4) \\ R(4,4) &= b_{4,4} + M_{rb}(4,4) & G(4,4) &= b_{4,4} + M_{gb}(4,4). \end{aligned} \tag{3.8}$$

3.1.4 Gradient-Based

Gradient-based CFA interpolation [36] is a multi-step method that preserves edge information by trying to prevent interpolation across edges. The green channel, $G(\cdot)$, is first estimated using an adaptive interpolation technique. For example, in order to estimate the green values at location $(3,3)$ and $(4,4)$, Figure 3.1, derivative estimators along the horizontal and vertical directions are first computed:

$$\begin{aligned} H_{3,3} &= |(r_{3,1} + r_{3,5})/2 - r_{3,3}| & V_{3,3} &= |(r_{1,3} + r_{5,3})/2 - r_{3,3}| \\ H_{4,4} &= |(b_{4,2} + b_{4,6})/2 - b_{4,4}| & V_{4,4} &= |(b_{2,4} + b_{6,4})/2 - b_{4,4}|. \end{aligned} \tag{3.9}$$

These estimators are referred to as classifiers. The estimated green values are then given by:

$$G(3,3) = \begin{cases} (g_{3,2} + g_{3,4})/2 & H_{3,3} < V_{3,3} \\ (g_{2,3} + g_{4,3})/2 & H_{3,3} > V_{3,3} \\ (g_{2,3} + g_{3,2} + g_{3,4} + g_{4,3})/4 & H_{3,3} = V_{3,3} \end{cases}$$
$$G(4,4) = \begin{cases} (g_{4,3} + g_{4,5})/2 & H_{4,4} < V_{4,4} \\ (g_{3,4} + g_{5,4})/2 & H_{4,4} > V_{4,4} \\ (g_{3,4} + g_{4,3} + g_{4,5} + g_{5,4})/4 & H_{4,4} = V_{4,4}. \end{cases} \tag{3.10}$$

The missing red samples are estimated by taking the difference image between the $\tilde{R}(\cdot)$ and $G(\cdot)$ channels, bilinearly interpolating this difference, then adding back $G(\cdot)$. For example, the missing red samples at $(3,4)$, $(4,3)$, and $(4,4)$, Figure 3.1, are given by:

$$R(3,4) \;=\; \frac{1}{2}\Big[\big(r_{3,3} - G(3,3)\big) + \big(r_{3,5} - G(3,5)\big)\Big] + G(3,4) \tag{3.11}$$

[1]The choice of the median filter size is important for the perceptual quality of CFA interpolated images.

$$R(4,3) = \frac{1}{2}\Big[\big(r_{3,3} - G(3,3)\big) + \big(r_{5,3} - G(5,3)\big)\Big] + G(4,3) \tag{3.12}$$

$$R(4,4) = \frac{1}{4}\Big[\big(r_{3,3} - G(3,3)\big) + \big(r_{3,5} - G(3,5)\big) + \big(r_{5,3} - G(5,3)\big) + \big(r_{5,5} - G(5,5)\big)\Big] + G(4,4). \tag{3.13}$$

The missing blue samples are estimated in a similar manner as the red samples.

3.1.5 Adaptive Color Plane

Adaptive color plane CFA interpolation [28] is a modification of the gradient-based interpolation method that includes second-order derivative estimates. The green channel, $G(\cdot)$, is first estimated using adaptive interpolation. For example, in order to estimate the green values at $(3,3)$ and $(4,4)$, Figure 3.1, horizontal and vertical classifiers are computed as the sum between first-order derivative estimates from the luminance (green) channel, and second-order derivative estimates from the chrominance (red and blue) channels:

$$\begin{aligned} H_{3,3} &= |g_{3,2} - g_{3,4}| + |-r_{3,1} + 2r_{3,3} - r_{3,5}| \\ V_{3,3} &= |g_{2,3} - g_{4,3}| + |-r_{1,3} + 2r_{3,3} - r_{5,3}| \\ H_{4,4} &= |g_{4,3} - g_{4,5}| + |-b_{4,2} + 2b_{4,4} - b_{4,6}| \\ V_{4,4} &= |g_{3,4} - g_{5,4}| + |-b_{2,4} + 2b_{4,4} - b_{6,4}|\,. \end{aligned} \tag{3.14}$$

Using these classifiers, the estimated green values are given by:

$$G(3,3) = \begin{cases} (g_{3,2} + g_{3,4})/2 + (-r_{3,1} + 2r_{3,3} - r_{3,5})/4 & H_{3,3} < V_{3,3} \\ (g_{2,3} + g_{4,3})/2 + (-r_{1,3} + 2r_{3,3} - r_{5,3})/4 & H_{3,3} > V_{3,3} \\ (g_{2,3} + g_{3,2} + g_{3,4} + g_{4,3})/4 + & \\ \qquad (-r_{1,3} - r_{3,1} + 4r_{3,3} - r_{3,5} - r_{5,3})/8 & H_{3,3} = V_{3,3} \end{cases}$$

$$G(4,4) = \begin{cases} (g_{4,3} + g_{4,5})/2 + (-b_{4,2} + 2b_{4,4} - b_{4,6})/4 & H_{4,4} < V_{4,4} \\ (g_{3,4} + g_{5,4})/2 + (-b_{2,4} + 2b_{4,4} - b_{6,4})/4 & H_{4,4} > V_{4,4} \\ (g_{3,4} + g_{4,3} + g_{4,5} + g_{5,4})/4 + & \\ \qquad (-b_{2,4} - b_{4,2} + 4b_{4,4} - b_{4,6} - b_{6,4})/8 & H_{4,4} = V_{4,4}. \end{cases} \tag{3.15}$$

The estimation of the missing chrominance samples depends on their location in the CFA. At locations where the nearest chrominance samples are in the same row or column, a non-adaptive procedure with second-order correction terms is employed. For example, red and blue samples at $(3,4)$ and $(4,3)$ are given by:

$$R(3,4) = \frac{r_{3,3} + r_{3,5}}{2} + \frac{-G(3,3) + 2G(3,4) - G(3,5)}{2} \tag{3.16}$$

$$B(3,4) = \frac{b_{2,4} + b_{4,4}}{2} + \frac{-G(2,4) + 2G(3,4) - G(4,4)}{2} \tag{3.17}$$

$$R(4,3) = \frac{r_{3,3}+r_{5,3}}{2} + \frac{-G(3,3)+2G(4,3)-G(5,3)}{2} \tag{3.18}$$

$$B(4,3) = \frac{b_{4,2}+b_{4,4}}{2} + \frac{-G(4,2)+2G(4,3)-G(4,4)}{2}. \tag{3.19}$$

At locations where the nearest chrominance neighbors are diagonally located, an adaptive procedure is employed. For example, in order to estimate the missing chrominance samples at $(3,3)$ and $(4,4)$, classifiers based on derivative estimates along the first and second diagonals are first estimated:

$$\begin{aligned} D^1_{3,3} &= |b_{2,2}-b_{4,4}| + |-G(2,2)+2G(3,3)-G(4,4)| \\ D^2_{3,3} &= |b_{2,4}-b_{4,2}| + |-G(2,4)+2G(3,3)-G(4,2)| \\ D^1_{4,4} &= |r_{3,3}-r_{5,5}| + |-G(3,3)+2G(4,4)-G(5,5)| \\ D^2_{4,4} &= |r_{3,5}-r_{5,3}| + |-G(3,5)+2G(4,4)-G(5,3)|, \end{aligned} \tag{3.20}$$

then using these classifiers, the missing red and blue values are given by:

$$B(3,3) = \begin{cases} (b_{2,4}+b_{4,2})/2 + (-G(2,4)+2G(3,3)-G(4,2))/2 & D^1_{3,3} > D^2_{3,3} \\ (b_{2,2}+b_{4,4})/2 + (-G(2,2)+2G(3,3)-G(4,4))/2 & D^1_{3,3} < D^2_{3,3} \\ (b_{2,2}+b_{2,4}+b_{4,2}+b_{4,4})/4 + & \\ \quad (-G(2,2)-G(2,4)+4G(3,3)-G(4,2)-G(4,4))/4 & D^1_{3,3} = D^2_{3,3} \end{cases}$$

$$R(4,4) = \begin{cases} (r_{3,5}+r_{5,3})/2 + (-G(3,5)+2G(4,4)-G(5,3))/2 & D^1_{4,4} > D^2_{4,4} \\ (r_{3,3}+r_{5,5})/2 + (-G(3,3)+2G(4,4)-G(5,5))/2 & D^1_{4,4} < D^2_{4,4} \\ (r_{3,3}+r_{3,5}+r_{5,3}+r_{5,5})/4 + & \\ \quad (-G(3,3)-G(3,5)+4G(4,4)-G(5,3)-G(5,5))/4 & D^1_{4,4} = D^2_{4,4}. \end{cases}$$

3.1.6 Threshold-Based Variable Number of Gradients

Threshold-based variable number of gradients CFA interpolation [9] is a more complex demosaicking method. At each pixel location, eight gradient estimates are first computed using samples from a 5×5 neighborhood of the CFA image. These estimates are computed slightly differently depending on the color sample at the pixel under consideration: chrominance (red and blue) or luminance (green). For example, the eight gradient estimates at location $(3,3)$, Figure 3.1, are given by:

$$N_{3,3} = |g_{2,3}-g_{4,3}| + |r_{1,3}-r_{3,3}| + \frac{|b_{2,2}-b_{4,2}|}{2} + \frac{|b_{2,4}-b_{4,4}|}{2} + \frac{|g_{1,2}-g_{3,2}|}{2} + \frac{|g_{1,4}-g_{3,4}|}{2} \tag{3.21}$$

$$E_{3,3} = |g_{3,2}-g_{3,4}| + |r_{3,3}-r_{3,5}| + \frac{|b_{2,2}-b_{2,4}|}{2} + \frac{|b_{4,2}-b_{4,4}|}{2} + \frac{|g_{2,3}-g_{2,5}|}{2} + \frac{|g_{4,3}-g_{4,5}|}{2} \tag{3.22}$$

$$S_{3,3} = |g_{2,3} - g_{4,3}| + |r_{3,3} - r_{5,3}| + \frac{|b_{2,2} - b_{4,2}|}{2} + \frac{|b_{2,4} - b_{4,4}|}{2} + \frac{|g_{3,2} - g_{5,2}|}{2} + \frac{|g_{3,4} - g_{5,4}|}{2} \quad (3.23)$$

$$W_{3,3} = |g_{3,2} - g_{3,4}| + |r_{3,1} - r_{3,3}| + \frac{|b_{2,2} - b_{2,4}|}{2} + \frac{|b_{4,2} - b_{4,4}|}{2} + \frac{|g_{2,1} - g_{2,3}|}{2} + \frac{|g_{4,1} - g_{4,3}|}{2} \quad (3.24)$$

$$NE_{3,3} = |b_{2,4} - b_{4,2}| + |r_{1,5} - r_{3,3}| + \frac{|g_{2,3} - g_{3,2}|}{2} + \frac{|g_{3,4} - g_{4,3}|}{2} + \frac{|g_{1,4} - g_{2,3}|}{2} + \frac{|g_{2,5} - g_{3,4}|}{2} \quad (3.25)$$

$$SE_{3,3} = |b_{2,2} - b_{4,4}| + |r_{3,3} - r_{5,5}| + \frac{|g_{2,3} - g_{3,4}|}{2} + \frac{|g_{3,2} - g_{4,3}|}{2} + \frac{|g_{3,4} - g_{4,5}|}{2} + \frac{|g_{4,3} - g_{5,4}|}{2} \quad (3.26)$$

$$NW_{3,3} = |b_{2,2} - b_{4,4}| + |r_{1,1} - r_{3,3}| + \frac{|g_{1,2} - g_{2,3}|}{2} + \frac{|g_{2,1} - g_{3,2}|}{2} + \frac{|g_{2,3} - g_{3,4}|}{2} + \frac{|g_{3,2} - g_{4,3}|}{2} \quad (3.27)$$

$$SW_{3,3} = |b_{2,4} - b_{4,2}| + |r_{5,1} - r_{3,3}| + \frac{|g_{2,3} - g_{3,2}|}{2} + \frac{|g_{3,4} - g_{4,3}|}{2} + \frac{|g_{3,2} - g_{4,1}|}{2} + \frac{|g_{4,3} - g_{5,2}|}{2}, \quad (3.28)$$

and the gradients at location $(3, 4)$, Figure 3.1, are given by:

$$N_{3,4} = |b_{2,4} - b_{4,4}| + |g_{1,4} - g_{3,4}| + \frac{|g_{2,3} - g_{4,3}|}{2} + \frac{|g_{2,5} - g_{4,5}|}{2} + \frac{|r_{1,3} - r_{3,3}|}{2} + \frac{|r_{1,5} - r_{3,5}|}{2} \quad (3.29)$$

$$E_{3,4} = |r_{3,3} - r_{3,5}| + |g_{3,4} - g_{3,6}| + \frac{|g_{2,3} - g_{2,5}|}{2} + \frac{|g_{4,3} - g_{4,5}|}{2} + \frac{|b_{2,4} - b_{2,6}|}{2} + \frac{|b_{4,4} - b_{4,6}|}{2} \quad (3.30)$$

$$S_{3,4} = |b_{2,4} - b_{4,4}| + |g_{3,4} - g_{5,4}| + \frac{|g_{2,3} - g_{4,3}|}{2} + \frac{|g_{2,5} - g_{4,5}|}{2} + \frac{|r_{3,3} - r_{5,3}|}{2} + \frac{|r_{3,5} - r_{5,5}|}{2} \quad (3.31)$$

$$W_{3,4} = |r_{3,3} - r_{3,5}| + |g_{3,2} - g_{3,4}| + \frac{|g_{2,3} - g_{2,5}|}{2} + \frac{|g_{4,3} - g_{4,5}|}{2} + \frac{|b_{2,2} - b_{2,4}|}{2} + \frac{|b_{4,2} - b_{4,4}|}{2} \quad (3.32)$$

$$NE_{3,4} = |g_{2,5} - g_{4,3}| + |g_{1,6} - g_{3,4}| + |r_{1,5} - r_{3,3}| + |b_{2,6} - b_{4,4}| \quad (3.33)$$

$$SE_{3,4} = |g_{2,3} - g_{4,5}| + |g_{3,4} - g_{5,6}| + |b_{2,4} - b_{4,6}| + |r_{3,3} - r_{5,5}| \quad (3.34)$$

$$NW_{3,4} = |g_{2,3} - g_{4,5}| + |g_{1,2} - g_{3,4}| + |b_{2,2} - b_{4,4}| + |r_{1,3} - r_{3,5}| \quad (3.35)$$

$$SW_{3,4} = |g_{2,5} - g_{4,3}| + |g_{3,4} - g_{5,2}| + |b_{2,4} - b_{4,2}| + |r_{3,5} - r_{5,3}| \ . \quad (3.36)$$

Let $\mathcal{G}_{x,y}$ denote the set of gradient estimates at (x, y):

$$\mathcal{G}_{x,y} = \{N_{x,y}, S_{x,y}, E_{x,y}, W_{x,y}, NE_{x,y}, NW_{x,y}, SE_{x,y}, SW_{x,y}\}. \quad (3.37)$$

Eight average red, green, and blue values, one for each gradient direction, are then computed at each pixel location. At location $(3, 3)$, for example, the average values corresponding to the different gradient regions are given by:

$$\begin{array}{lll}
R^N_{3,3} = (r_{1,3} + r_{3,3})/2 & G^N_{3,3} = g_{2,3} & B^N_{3,3} = (b_{2,2} + b_{2,4})/2 \\
R^E_{3,3} = (r_{3,3} + r_{3,5})/2 & G^E_{3,3} = g_{3,4} & B^E_{3,3} = (b_{2,4} + b_{4,4})/2 \\
R^S_{3,3} = (r_{3,3} + r_{5,3})/2 & G^S_{3,3} = g_{4,3} & B^S_{3,3} = (b_{4,2} + b_{4,4})/2 \\
R^W_{3,3} = (r_{3,1} + r_{3,3})/2 & G^W_{3,3} = g_{3,2} & B^W_{3,3} = (b_{2,2} + b_{4,2})/2 \\
R^{NE}_{3,3} = (r_{1,5} + r_{3,3})/2 & G^{NE}_{3,3} = (g_{1,4} + g_{2,3} + g_{2,5} + g_{3,4})/4 & B^{NE}_{3,3} = b_{2,4} \\
R^{SE}_{3,3} = (r_{3,3} + r_{5,5})/2 & G^{SE}_{3,3} = (g_{3,4} + g_{4,3} + g_{4,5} + g_{5,4})/4 & B^{SE}_{3,3} = b_{4,4} \\
R^{NW}_{3,3} = (r_{1,1} + r_{3,3})/2 & G^{NW}_{3,3} = (g_{1,2} + g_{2,1} + g_{2,3} + g_{3,2})/4 & B^{NW}_{3,3} = b_{2,2} \\
R^{SW}_{3,3} = (r_{3,3} + r_{5,1})/2 & G^{SW}_{3,3} = (g_{3,2} + g_{4,1} + g_{4,3} + g_{5,2})/4 & B^{SW}_{3,3} = b_{4,2}
\end{array}$$

From $\mathcal{G}_{x,y}$, Equation (3.37), a threshold $T_{x,y}$ is computed as:

$$T_{x,y} = k_1 \min \mathcal{G}_{x,y} + k_2(\max \mathcal{G}_{x,y} - \min \mathcal{G}_{x,y}), \quad (3.38)$$

where the values of k_1, k_2 are empirically chosen as 1.5 and 0.5, respectively. Next the previously computed color averages from regions whose gradient estimate is less than the threshold, are averaged to yield $R^{avg}_{x,y}$, $G^{avg}_{x,y}$, and $B^{avg}_{x,y}$. For example, at location $(3, 3)$ if the gradient estimates less than $T_{3,3}$ are $S_{3,3}$, $W_{3,3}$, $NE_{3,3}$, and $SW_{3,3}$, then:

$$\begin{aligned}
R^{avg}_{3,3} &= 1/4 \cdot (R^S_{3,3} + R^W_{3,3} + R^{NE}_{3,3} + R^{SW}_{3,3}) \\
G^{avg}_{3,3} &= 1/4 \cdot (G^S_{3,3} + G^W_{3,3} + G^{NE}_{3,3} + G^{SW}_{3,3}) \\
B^{avg}_{3,3} &= 1/4 \cdot (B^S_{3,3} + B^W_{3,3} + B^{NE}_{3,3} + B^{SW}_{3,3}).
\end{aligned} \quad (3.39)$$

The final estimates of the missing color samples at location $(3, 3)$ are then given by:

$$\begin{aligned}
G(3, 3) &= r_{3,3} + (G^{avg}_{3,3} - R^{avg}_{3,3}) \\
B(3, 3) &= r_{3,3} + (B^{avg}_{3,3} - R^{avg}_{3,3}).
\end{aligned} \quad (3.40)$$

The missing color samples at the other pixel locations are estimated in a similar manner.

3.2 Detecting CFA Interpolation

At each pixel location of a CFA interpolated color image, a single color sample is captured by the camera sensor, while the other two color samples are estimated from neighboring samples. As a result, a subset of samples, within a color channel, are correlated to their neighboring samples. Since the color filters in a CFA are typically arranged in a periodic pattern, these correlations are periodic. Consider, for example, the red channel, $R(x,y)$, that has been sampled on a Bayer array, Figure 3.1, then CFA interpolated using bilinear interpolation, Section 3.1.1. In this case, the red samples in the odd rows and even columns are the average of their closest horizontal neighbors, the red samples in the even rows and odd columns are the average of their closest vertical neighbors, and the red samples in the even rows and columns are the average of their closest diagonal neighbors:

$$\begin{aligned} R(2x+1,2y) &= \frac{R(2x+1,2y-1)}{2} + \frac{R(2x+1,2y+1)}{2} \\ R(2x,2y+1) &= \frac{R(2x-1,2y+1)}{2} + \frac{R(2x+1,2y+1)}{2} \\ R(2x,2y) &= \frac{R(2x-1,2y-1)}{4} + \frac{R(2x-1,2y+1)}{4} + \\ &\qquad \frac{R(2x+1,2y-1)}{4} + \frac{R(2x+1,2y+1)}{4}. \end{aligned} \tag{3.41}$$

Note that in this simple case, the estimated samples are perfectly correlated to their neighbors. As such a CFA interpolated image can be detected (in the absence of noise) by noticing, for example, that every other sample in every other row or column is perfectly correlated to its neighbors. At the same time, the non-interpolated samples are less likely to be correlated in precisely the same manner. Furthermore, it is likely that tampering will destroy these correlations, or that the splicing together of two images from different cameras will create inconsistent correlations across the composite image. As such, the presence, or lack, of correlations due to CFA interpolation can be used to authenticate an image, or expose it as a forgery.

We begin by assuming a simple linear model for the periodic correlations introduced by CFA interpolation. That is, each interpolated pixel is correlated to a weighted sum of pixels in a small neighborhood centered about itself. While perhaps overly simplistic when compared to the highly non-linear nature of most CFA interpolation algorithms, this simple model is both easy to parameterize and can reasonably approximate each of the CFA interpolation algorithms presented in Section 3.1. Note that most CFA algorithms estimate a missing color sample from neighboring samples in all three color channels. For simplicity, however, we ignore these inter-channel correlations and treat each color channel independently.

If the specific form of the correlations are known (i.e., the parameters of the linear model), then it would be straightforward to determine which samples are correlated to their neighbors. On the other hand, if the samples correlated to their neighbors are known, the specific form of the correlations could be easily determined. In practice, of course, neither are known. To simultaneously

estimate both we employ the expectation-maximization[2] (EM) algorithm [14], as described below.

3.2.1 Expectation-Maximization Algorithm

Let $f(\cdot,\cdot)$ denote a color channel (red, green, or blue) of a CFA interpolated image. We begin by assuming that each sample in $f(\cdot,\cdot)$ belongs to one of two models:

M_1, if the sample is linearly correlated to its neighbors, i.e., is generated by the linear model:

$$f(x,y) = \sum_{u,v=-N}^{N} \alpha_{u,v} f(x+u, y+v) + n(x,y), \tag{3.42}$$

where the model parameters are given by $\vec{\alpha} = \{\alpha_{u,v} | -N \leq u, v \leq N\}$ (N is an integer and $\alpha_{0,0} = 0$), the linear coefficients (strung out in vector form) that embody the specific form of the correlations, with $\alpha_{0,0} = 0$. In the above equation, $n(x,y)$ denote independent, and identically distributed (i.i.d.) samples drawn from a Gaussian distribution with zero mean and unknown variance σ^2.

M_2, if the sample is not correlated to its neighbors, i.e., is generated by an outlier process.

The expectation-maximization algorithm (EM) is a two-step iterative algorithm: (1) in the E-step the probability of each sample belonging to each model is estimated; and (2) in the M-step the specific form of the correlations between samples is estimated. More specifically, in the E-step, the probability of each sample of $f(x,y)$ belonging to model M_1 is estimated using Bayes' rule:

$$\Pr\{f(x,y) \in M_1 \mid f(x,y)\} = \frac{\Pr\{f(x,y) \mid f(x,y) \in M_1\} \Pr\{f(x,y) \in M_1\}}{\sum_{i=1}^{2} \Pr\{f(x,y) \mid f(x,y) \in M_i\} \Pr\{f(x,y) \in M_i\}}, \tag{3.43}$$

where the prior probabilities $\Pr\{f(x,y) \in M_1\}$ and $\Pr\{f(x,y) \in M_2\}$ are assumed to be equal to $1/2$. The probability of observing a sample $f(x,y)$ knowing it was generated from model M_1 is given by:

$$\Pr\{f(x,y) \mid f(x,y) \in M_1\} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma^2}\left(f(x,y) - \sum_{u,v=-N}^{N} \alpha_{u,v} f(x+u, y+v)\right)^2\right]. \tag{3.44}$$

[2] A short tutorial on the general EM algorithm, and an example of its application to a mixture models problem are presented in Appendix A.

The variance, σ^2, of this Gaussian distribution is estimated in the M-step (see Section 3.5). A uniform distribution is assumed for the probability of observing a sample generated by the outlier model, M_2, i.e., $\Pr\{f(x,y) \mid f(x,y) \in M_2\}$ is equal to the inverse of the range of possible values of $f(x,y)$. Note that the E-step requires an estimate of the coefficients $\vec{\alpha}$, which on the first iteration is chosen randomly. In the M-step, a new estimate of $\vec{\alpha}$ is computed using weighted least squares, i.e., minimizing the following quadratic error function:

$$E(\vec{\alpha}) \;=\; \sum_{x,y} w(x,y) \left(f(x,y) - \sum_{u,v=-N}^{N} \alpha_{u,v} f(x+u, y+v) \right)^2 , \tag{3.45}$$

where the weights $w(x,y) \equiv \Pr\{f(x,y) \in M_1 \mid f(x,y)\}$, Equation (3.43). This error function is minimized by computing the gradient with respect to $\vec{\alpha}$, setting this gradient equal to zero, and solving the resulting linear system of equations. Setting equal to zero the partial derivative with respect to one of the coefficients, $\alpha_{s,t}$, yields:

$$\begin{gathered}
\frac{\partial E}{\partial \alpha_{s,t}} = 0 \\
-2 \sum_{x,y} w(x,y) f(x+s, y+t) \left(f(x,y) - \sum_{u,v=-N}^{N} \alpha_{u,v} f(x+u, y+v) \right) = 0 \\
\sum_{x,y} w(x,y) f(x+s, y+t) \sum_{u,v=-N}^{N} \alpha_{u,v} f(x+u, y+v) = \\
\sum_{x,y} w(x,y) f(x+s, y+t) f(x,y).
\end{gathered} \tag{3.46}$$

Re-ordering the terms on the left-hand side yields:

$$\begin{aligned}
\sum_{u,v=-N}^{N} \alpha_{u,v} \left(\sum_{x,y} w(x,y) f(x+s, y+t) f(x+u, y+v) \right) = \\
\sum_{x,y} w(x,y) f(x+s, y+t) f(x,y) \; .
\end{aligned} \tag{3.47}$$

This process is repeated for each component, $\alpha_{s,t}$, of $\vec{\alpha}$, to yield a system of linear equations that can be solved using standard techniques.

The E-step and the M-step are iteratively executed until a stable estimate of $\vec{\alpha}$ is achieved — see Section 3.5 for a detailed algorithm. The final $\vec{\alpha}$ has the property that it maximizes the likelihood of the observed samples[3] — see Appendix A for more details on maximum likelihood estimation and EM.

[3]The EM algorithm is proved to always converge to a stable estimate that is a local maximum of the log-likelihood of the observed data, [14] and Appendix A.

3.3 Results

We collected one hundred images, fifty of resolution 512×512, and fifty of resolution 1024×1024. Each of these images were cropped from a smaller set of twenty 1600×1200 images taken with a Nikon Coolpix 950 camera, and twenty 3034×2024 images taken with a Nikon D100 camera. The Nikon Coolpix 950 employs a four-filter (yellow, cyan, magenta, green) CFA, and was set to store the images in uncompressed TIFF format. The Nikon D100 camera employs a Bayer array, and was set to store the images in RAW format. To avoid interference with the CFA interpolation of the cameras, each color channel of these images was independently blurred with a 3×3 binomial filter, and down-sampled by a factor of two in each direction. These down-sampled color images were then re-sampled onto a Bayer array, and CFA interpolated using the algorithms described in Section 3.1.

Shown in Figures 3.2 and 3.3 are the results of running the EM algorithm, Section 3.2.1, on eight color images that were CFA interpolated using the algorithms presented in Section 3.1, and on two images where the CFA correlations were destroyed by down-sampling. The parameters of the EM algorithm were fixed throughout at $N = 1$, $\sigma_0 = 0.0075$, and $p_0 = 1/256$ (see Section 3.5). Shown in the left column are the images, in the middle column the probability maps from the green channel (the red and blue channels yield similar results), and in the right column the magnitude of the Fourier transforms of the probability maps[4]. Note that, although the periodic patterns in the probability maps are not visible at this reduced scale, they are particularly salient in the Fourier domain in the form of localized peaks. Note also that these peaks do not appear in the images that are not CFA interpolated (last row of Figures 3.2 and 3.3).

While the probability maps obtained from the EM algorithm can be employed to detect if a color image is the result of a CFA interpolation algorithm, the linear coefficients, $\vec{\alpha}$, returned by the EM algorithm can be used to distinguish between different CFA interpolation techniques. Let $\vec{\alpha}$ denote the 24-D vector of linear coefficients (i.e., with a 3×3 neighborhood size, $N = 1$, each of the three color channels has 8 coefficients). Using principal component analysis (PCA) [15], we have computed the principal axes of the 800, 24-D vectors obtained from running the EM algorithm on 100 images CFA interpolated with each of the algorithms described in Section 3.1. Shown in Figure 3.4 are the projections onto the first two principal axes. Note how points corresponding to different CFA interpolation algorithms tend to cluster. Even in this two-dimensional space, these clusters are, in some cases, clearly separable.

While the different interpolation methods are not perfectly separable in the low-dimensional PCA space, they are pairwise separable in their original 24-D space. Specifically, a Linear Discriminant Analysis (LDA) [19] on all pairs of interpolation algorithms yields nearly perfect separation. For each of the twenty-eight distinct pairs of algorithms we trained a LDA classifier. Using 90% of the data for training and 10% for testing, the average testing accuracies over 10 random train-

[4]For display purposes, the probability maps were up-sampled by a factor of two before Fourier transforming. The periodic patterns introduced by CFA interpolation have energy in the highest horizontal, vertical and diagonal frequencies, which corresponds to localized frequency peaks adjacent to the image edges. Up-sampling by a factor of two shrinks the support of a probability map's spectrum, and shifts these peaks into the mid-frequencies, where they are easier to see. Also for display purposes, the Fourier transforms of the probability maps were high-pass filtered, blurred, scaled to fill the range $[0, 1]$, and gamma corrected with an exponent of 2.0.

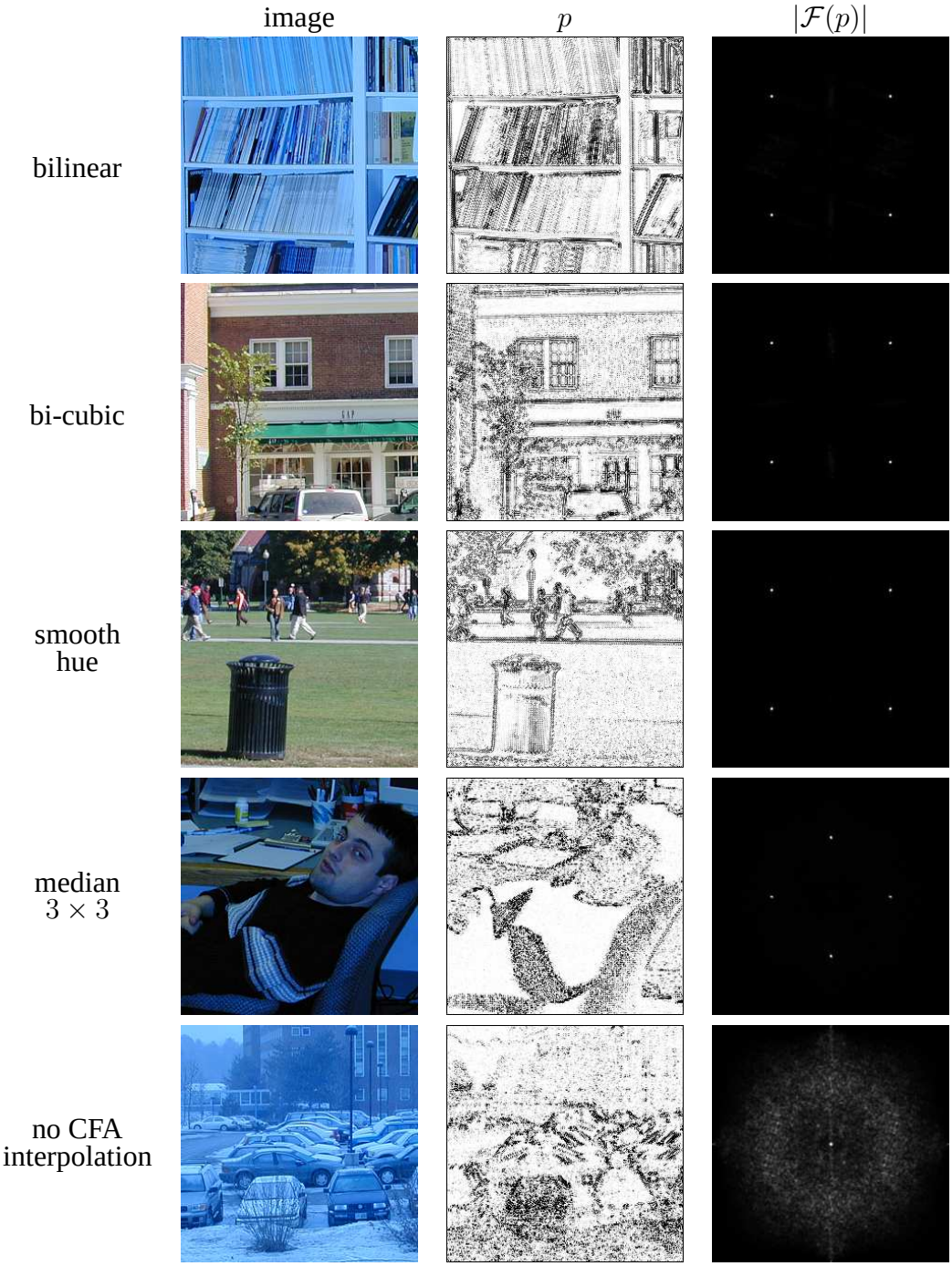

Figure 3.2: Shown in each row is an image interpolated with the specifi ed algorithm, the probability map of the green channel as output by the EM algorithm, and the magnitude of the Fourier transform of the probability map. Note the periodic correlations in the CFA interpolated images (peaks in $|\mathcal{F}(p)|$) and the lack of such correlations in the non-CFA interpolated image (last row).

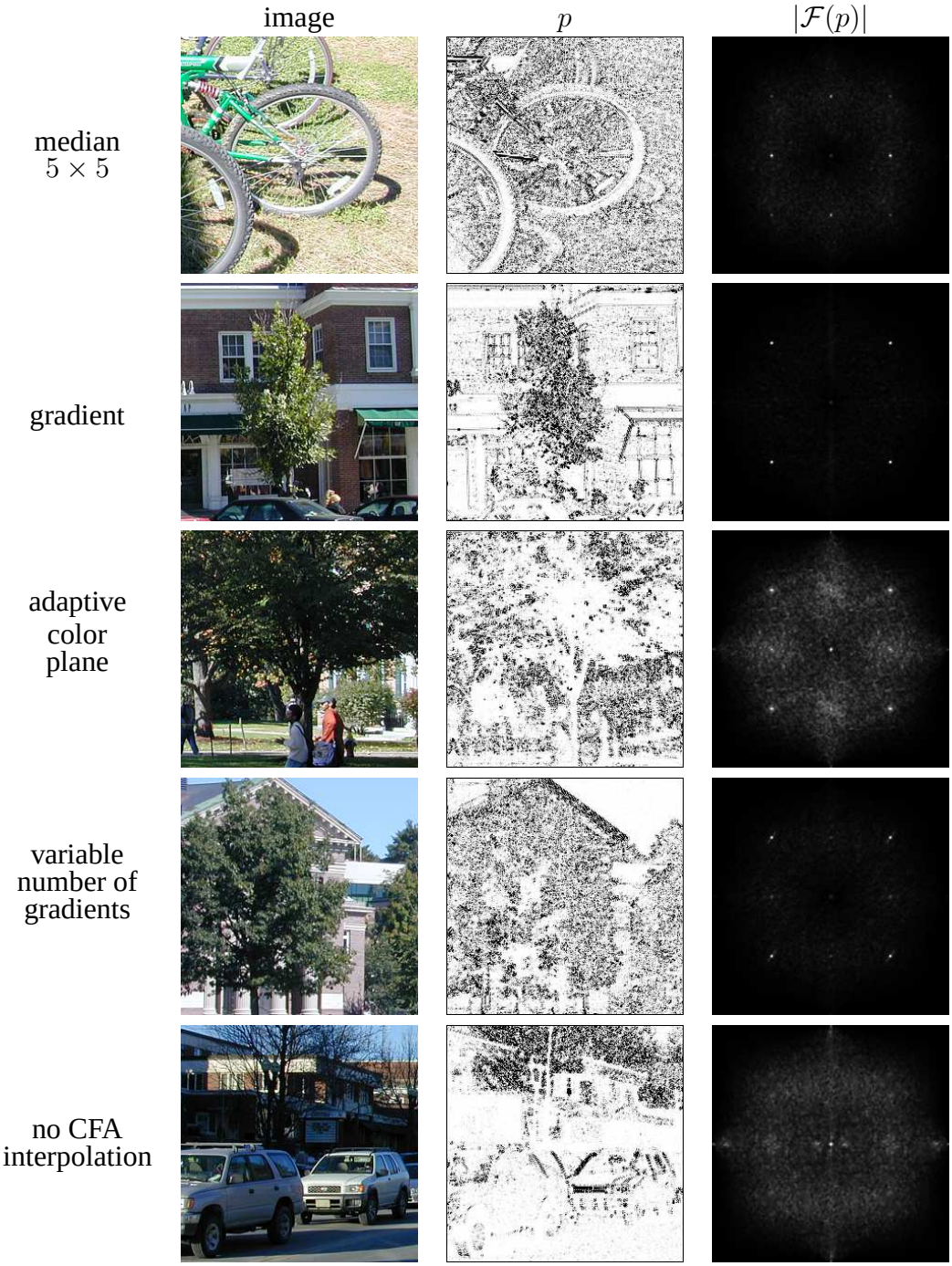

Figure 3.3: Shown in each row is an image interpolated with the specifi ed algorithm, the probability map of the green channel as output by the EM algorithm, and the magnitude of the Fourier transform of the probability map. Note the periodic correlations in the CFA interpolated images (peaks in $|\mathcal{F}(p)|$) and the lack of such correlations in the non-CFA interpolated image (last row).

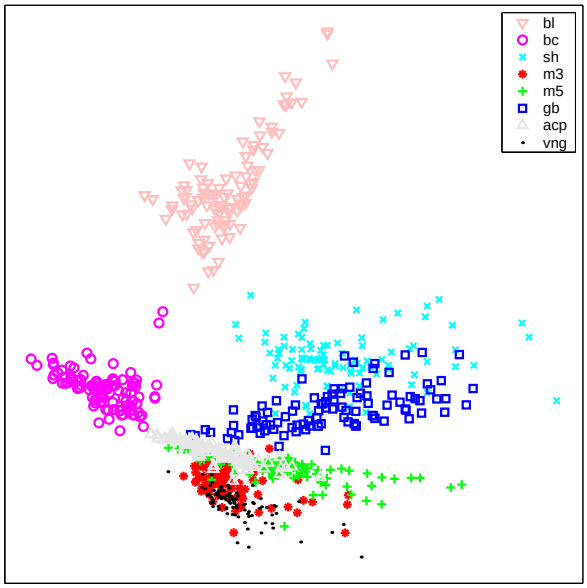

Figure 3.4: The estimated interpolation coeffi cients from 100 images CFA interpolated with eight different algorithms are projected onto a 2-D subspace. Even in this reduced space, the algorithms are, in some cases, clearly separable.

ing/testing splits and over all pairs of algorithms are shown in the Table 3.1. The average over all pairs of algorithms was 97%, and the minimum testing accuracy was 87% and was obtained when separating the 3×3 median filter and the variable number of gradients algorithms. Since digital cameras typically employ different CFA interpolation algorithms, this result could be used to determine if an image was *not* taken with a specific digital camera.

3.3.1 Detecting Localized Tampering

Since it is likely that tampering will destroy the periodic CFA correlations, it may be possible to detect and localize tampering in any portion of an image. To illustrate this, we begin with three 1024×1024 images, taken with a Nikon D100 digital camera and saved in RAW format, Figure 3.5. To simulate tampering, each color channel of these images (initially interpolated using the adaptive color plane technique) was blurred with a 3×3 binomial filter and down-sampled by a factor of two in order to destroy the CFA correlations. The 512×512 down-sampled images were then re-sampled on a Bayer array and CFA interpolated. Next, composite images, 512×512 pixels in size, were created by splicing, in varying proportions ($1/4$, $1/2$, and $3/4$), a non-CFA interpolated image and the same CFA interpolated image. Shown in Figure 3.6 are the results of running EM on these composite images — each row corresponds to a different interpolation algorithm: from top to bottom, bi-cubic, Section 3.1.1, gradient-based, Section 3.1.4, and variable number of gradients, Section 3.1.6. Shown in the top row are the probability maps of the red channel. Note that these probability maps clearly reveal the presence of two distinct regions. Also

–	bl	bc	sh	m3	m5	gb	acp	vng
bl	–	100%	100%	100%	100%	100%	100%	100%
bc	100%	–	100%	100%	100%	100%	99.5%	100%
sh	100%	100%	–	100%	100%	97.5%	100%	100%
m3	100%	100%	100%	–	97%	100%	98.5%	87%
m5	100%	100%	100%	97%	–	99.5%	94.5%	97.5%
gb	100%	100%	97.5%	100%	99.5%	–	99.5%	100%
acp	100%	99.5%	100%	98.5%	94.5%	99.5%	–	100%
vng	100%	100%	100%	87%	97.5%	100%	100%	–

Table 3.1: Shown in this table are the accuracies obtained using linear discriminant analysis (LDA) on pairs of CFA interpolation algorithms. Each accuracy value was computed using 100 images (90 for training, and 10 for testing), and 10 random training/testing splits. The different CFA interpolation algorithms were: bilinear (bl), bi-cubic (bc), smooth hue (sh), 3×3 median filter based (m3), 5×5 median filter based (m5), gradient based (gb), adaptive color plane (acp), and variable number of gradients (vng).

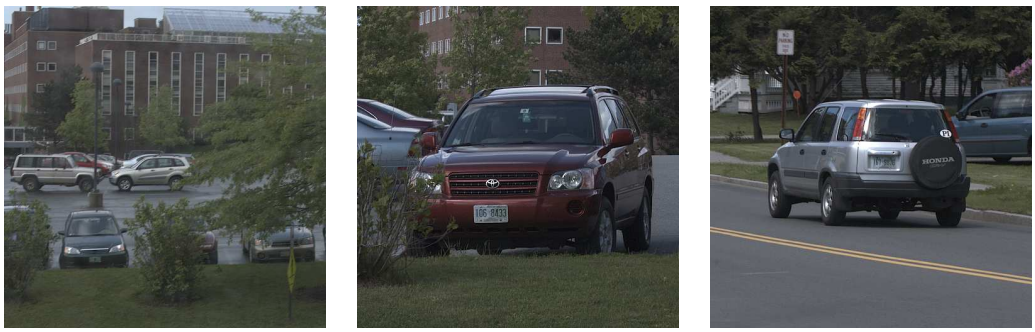

Figure 3.5: Shown are three images taken with a Nikon D100 digital camera in RAW format. See also Figure 3.6.

shown in the top row is the magnitude of the Fourier transforms of two windows[5], one from the non-CFA interpolated portion (right), and one from the CFA interpolated portion (left). Note the presence of localized frequency peaks in the CFA interpolated portion, and the lack of such peaks in the non-CFA interpolated portion. Shown in the middle and bottom rows are similar probability maps obtained from the green and blue channels of the other composite images, as well as the magnitude of the Fourier transforms of windows from the non-CFA and CFA interpolated portions of the maps.

Shown on top in Figure 3.7 are an original image (left) and a perceptually plausible forgery of the same image (right). The original image was CFA interpolated using bi-cubic interpolation. The

[5]For display purposes, the Fourier transforms of the probability maps were up-sampled by a factor of two, high-pass filtered, blurred to avoid aliasing artifacts, scaled to fill the range $[0, 1]$, and gamma corrected with an exponent of 2.0.

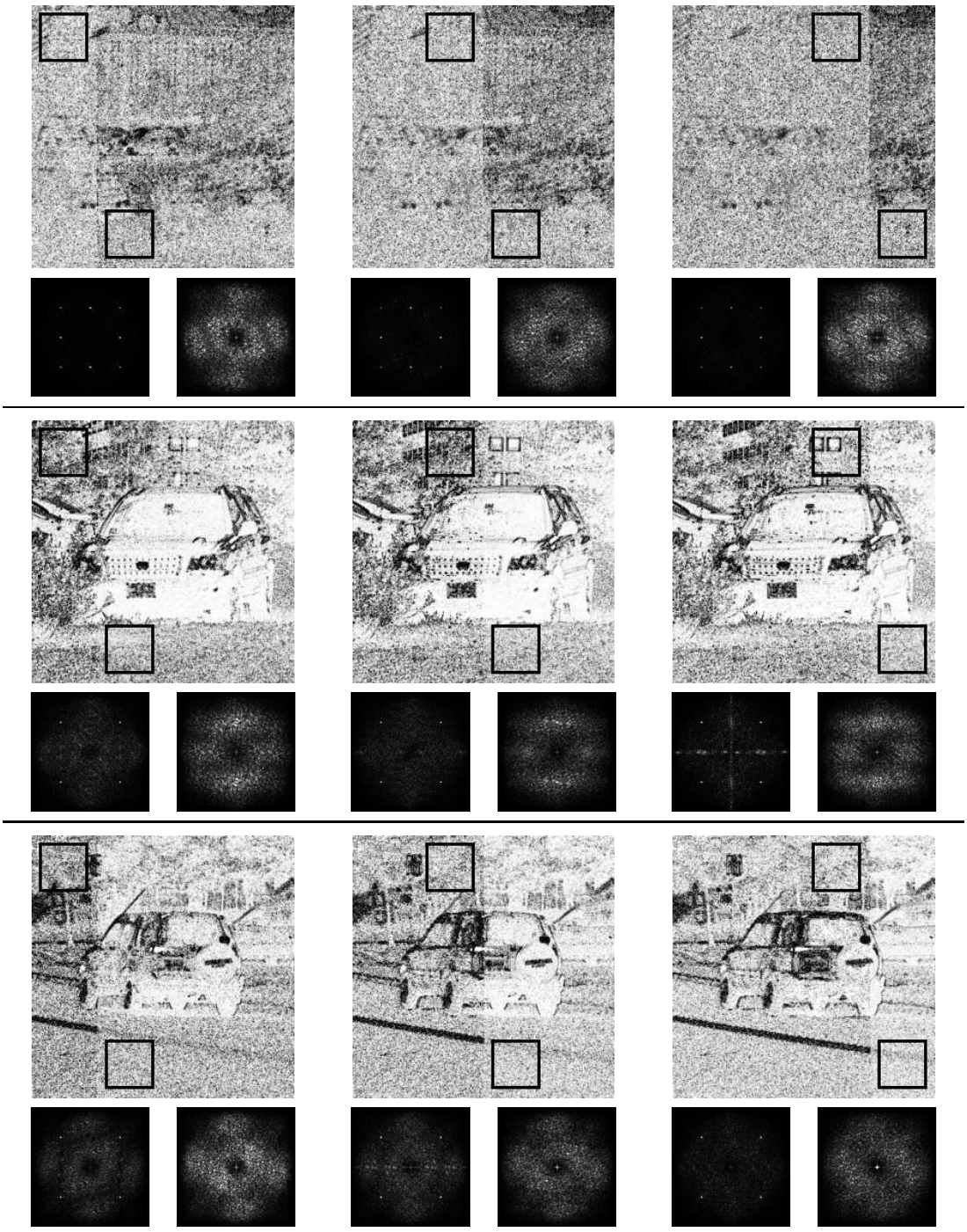

Figure 3.6: Shown are the probability maps of composite images obtained by splicing together CFA interpolated images (left portion of each image) and the same image without CFA interpolation (right portion) — the original images are shown in Figure 3.5. Also shown is the magnitude of the Fourier transforms of windows from the two regions. Note that windows from the CFA interpolated regions (left) have localized peaks in the Fourier domain, while the windows from the non-CFA interpolated regions (right) do not.

forged image was manipulated with *Adobe Photoshop*®, and the tampering consisted of hiding the damage on the car hood using air-brushing, smudging, blurring, and duplication. Shown on the bottom of Figure 3.7 are the probability map of the tampered image, and the magnitude of the Fourier transforms of two windows, one from a tampered portion (left), and one from an unadulterated portion (right). Note that even though the periodic pattern is not visible in the probability map, localized frequency peaks reveal its presence in the unadulterated region. Note also that the tampered region does not contain frequency peaks. Shown on top in Figure 3.8 is another example of an original image (left) and the same image after having been tampered with (right). The original image was CFA interpolated using the variable number of gradients algorithm. The tampered image was manipulated using *Adobe Photoshop*®, and the tampering consisting in overlaying a non-CFA interpolated image of a graffiti-painted wall over the facade of the building. Shown on the bottom of Figure 3.8 are the probability map of the tampered image, and the magnitude of the Fourier transforms of two windows one from a tampered portion (left), and one from an unadulterated portion (right). Note again how the presence of localized frequency peaks reveal the presence of the periodic pattern in the unadulterated region, while the absence of peaks in the other region reveals the tampering.

3.3.2 Sensitivity and Robustness

From a digital forensic point of view it is important to quantify the robustness and sensitivity of our detection technique. It is therefore necessary to devise a quantitative measure for the periodic correlations introduced by CFA interpolation. This is achieved by comparing the estimated probability maps of each color channel of a given image with synthetically generated probability maps. A synthetic map is generated for each color channel as follows:

$$
\begin{aligned}
s_r(x,y) &= \begin{cases} 0 & S(x,y) = r_{x,y} \\ 1 & \text{otherwise} \end{cases} \\
s_g(x,y) &= \begin{cases} 0 & S(x,y) = g_{x,y} \\ 1 & \text{otherwise} \end{cases} \\
s_b(x,y) &= \begin{cases} 0 & S(x,y) = b_{x,y} \\ 1 & \text{otherwise}, \end{cases}
\end{aligned}
\tag{3.48}
$$

where $S(x,y)$ is defined as in Section 3.1. Let $p_g(x,y)$ denote the probability map obtained from the green channel of an image. A similarity measure between $p_g(x,y)$ and $s_g(x,y)$ is computed as follows:

1. The probability map $p_g(\cdot)$ is Fourier transformed: $P_g(\omega_x,\omega_y) = \mathcal{F}(p_g(x,y))$.
2. The synthetic map $s_g(\cdot)$ is Fourier transformed: $S_g(\omega_x,\omega_y) = \mathcal{F}(s_g(x,y))$.
3. The measure of similarity between $p_g(x,y)$ and $s_g(x,y)$ is then given by:

$$
M(p_g,s_g) = \sum_{\omega_x,\omega_y} |P_g(\omega_x,\omega_y)| \cdot |S_g(\omega_x,\omega_y)|, \tag{3.49}
$$

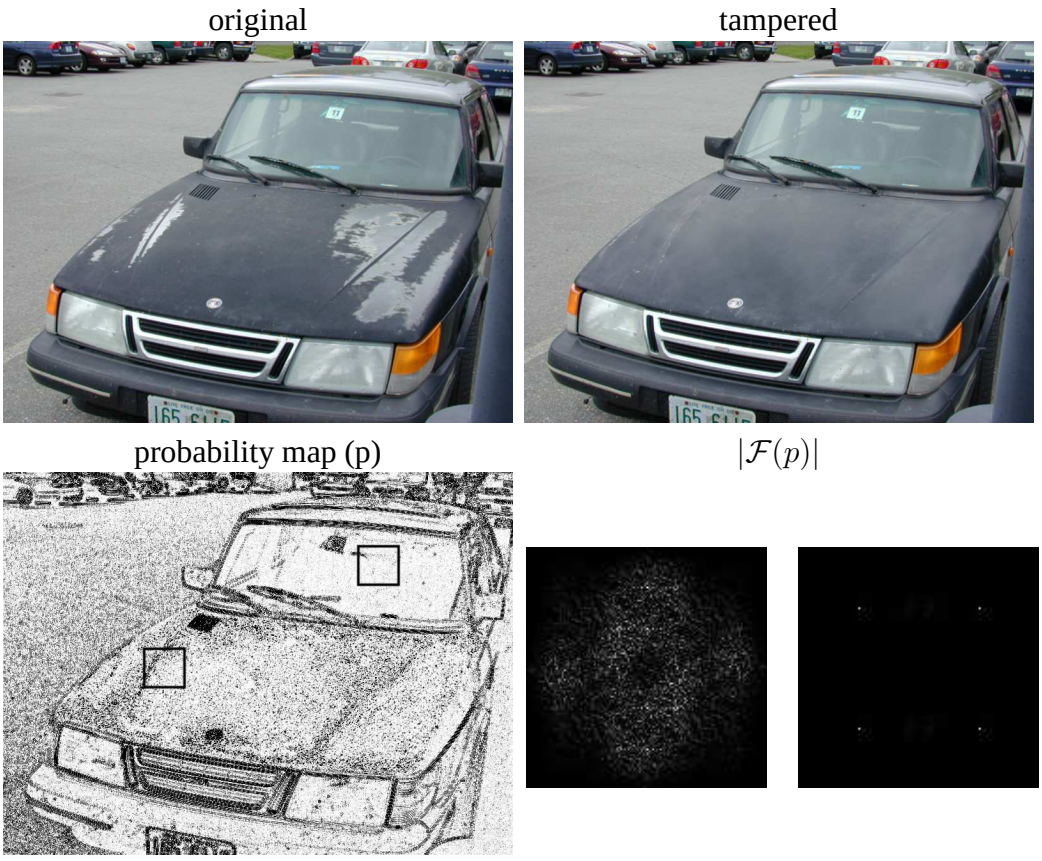

Figure 3.7: Shown on top are an original (left) and a tampered image (right). Shown on the bottom are the probability map of the tampered image's green channel, and the magnitude of the Fourier transform of two windows from the probability map corresponding to a tampered (left) and unadulterated (right) portion of the image. Note the lack of peaks in the tampered region signifying the absence of CFA interpolation.

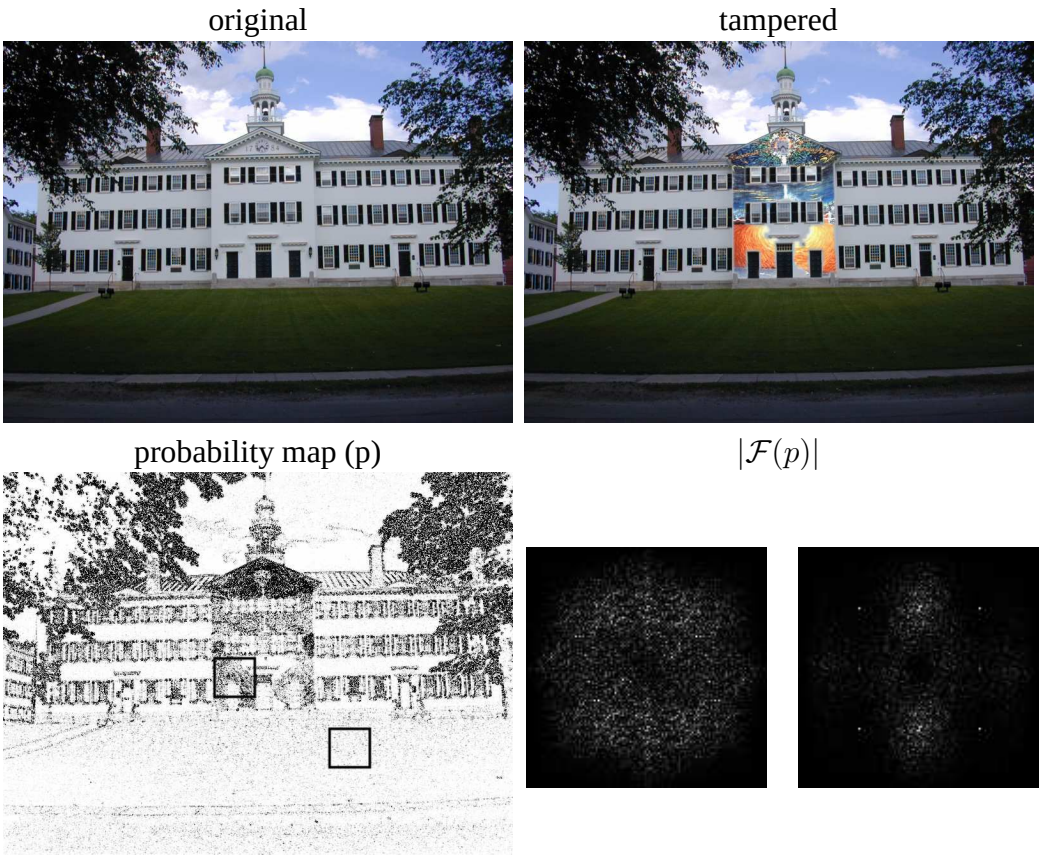

Figure 3.8: Shown on top are an original (left) and a tampered image (right). Shown on the bottom are the probability map of the tampered image's green channel, and the magnitude of the Fourier transform of two windows from the probability map corresponding to a tampered (left) and unadulterated (right) portion of the image. Note the lack of peaks in the tampered region signifying the absence of CFA interpolation.

where $|\cdot|$ denotes absolute value (note that this similarity measure is phase insensitive).

The similarity measures between the probability and synthetic maps of the red and blue channels, $M(p_r, s_r)$ and $M(p_b, s_b)$, are computed in a similar way. If the similarity measure, $M(p_g, s_g)$, between $p_g(x, y)$ and $s_g(x, y)$ is above a specified threshold, then a periodic pattern is assumed to be present in the probability map, and the channel is labeled as CFA interpolated. Thresholds are empirically determined for each channel to yield a 0% false positive rate, i.e., no color channel from non-CFA interpolated images is classified as CFA interpolated.

To simulate tampering, each image in our database is blurred and down-sampled in order to destroy the original CFA interpolation correlations. These images are then re-sampled onto a Bayer CFA and interpolated with each of the algorithms described in Section 3.1. Given a three-channel color image, we first compute the similarity measures between the probability maps of each color channel and the synthetic maps. The image is labeled as CFA interpolated if at least one of the three similarities is greater than the specified thresholds, or as tampered if all three similarities are less than the thresholds. Using this approach, we have obtained, with a 0% false positive rate (i.e., a non-CFA interpolated image is never misclassified), the following classification accuracies averaged over 100 images:

bilinear	bi-cubic	smooth hue	median (3×3)	median (5×5)	gradient-based	adaptive color plane	variable number of gradients
100%	100%	100%	99%	97%	100%	97%	100%

Counter Measures

To be useful in a forensic setting it is important for our detection method to be robust to counter-measures. We have tested the sensitivity of our method to simple counter attacks that may conceal traces of tampering: (1) lossy compression schemes (e.g., JPEG, GIF, and JPEG2000), (2) additive white Gaussian noise, and (3) non-linear point-wise gamma correction. Fifty images taken with the Nikon Coolpix 950 camera, were processed as described in the previous section in order to obtain CFA and non-CFA interpolated images. Shown in Figure 3.9 are the detection accuracies (with 0% false positives) for different CFA interpolation algorithms as a function of the JPEG compression quality. Note that these detection accuracies are close to 100% for quality factors greater than 96 (out of 100), and that they decrease gracefully when the quality factors are decreasing. This decrease in accuracy is expected, since the lossy JPEG compression introduces noise that interferes with the correlations introduced by CFA interpolation. The decrease in accuracy depends on the CFA interpolation algorithm, e.g., at quality 70, the accuracy for smooth hue CFA interpolation is 56%, while the accuracy for adaptive color plane CFA interpolation drops to 6%. We have also tested our detection method on images that have been 8-bit color indexed (GIF). The detection accuracies (with 0% false positives) were perfect (i.e., 100%) for all CFA interpolation algorithms, except for the adaptive color plane interpolation, where the accuracy was 96%. Tests with images compressed with the JPEG2000 scheme revealed a weakness in our approach. The quantization of wavelet coefficients employed by the JPEG2000 scheme, introduces periodic artifacts that are

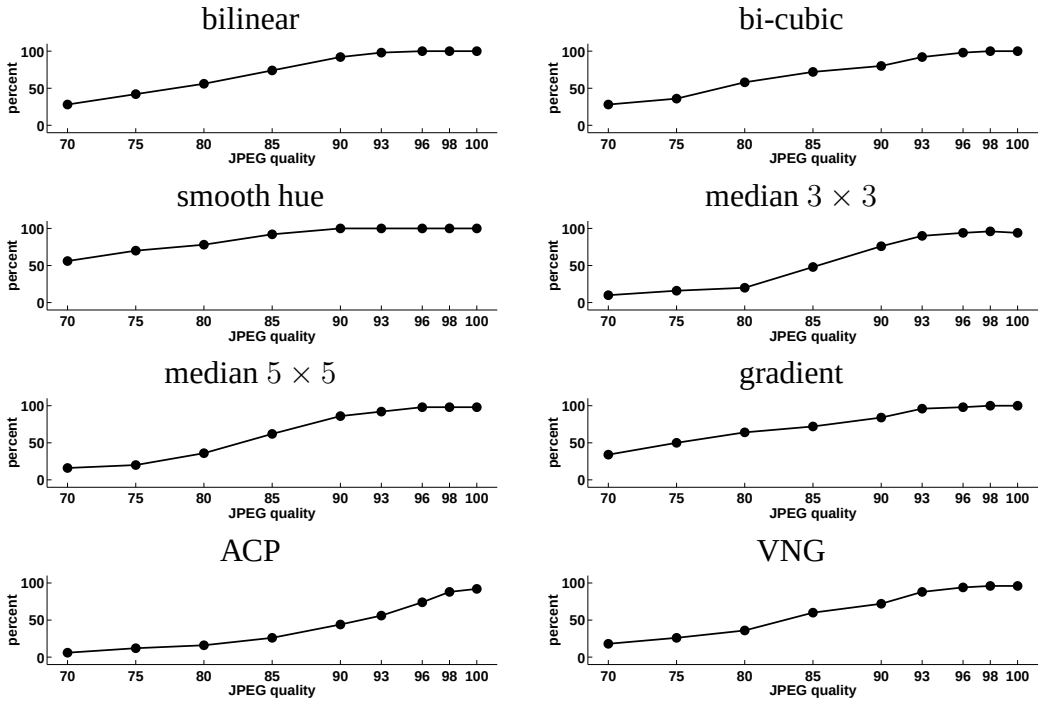

Figure 3.9: Detection accuracies (with 0% false positives) for different CFA interpolation algorithms as a function of JPEG compression quality. Each data point corresponds to the average detection accuracy over fifty images.

indistinguishable from those introduced by CFA interpolation. As a result, it is not possible to detect CFA interpolated images that have been subsequently JPEG2000 compressed.

Shown in Figure 3.10 are the detection accuracies (with 0% false positives) for different CFA interpolation algorithms as a function of the signal-to-noise ratio (SNR) of additive white Gaussian noise. In this case, the accuracies drop relatively slowly as the SNR decreases. For example, for a SNR of 18db, even non-linear methods achieve very good detection rates, e.g., 76% for the adaptive color plane technique, and 86% for the variable number of gradients technique. We have also tested our detection method on CFA interpolated images gamma corrected with exponents ranging from 0.5 to 1.5, in steps of 0.1. For all CFA interpolation techniques the accuracies (with 0% false positives) were either 100% or 98% (i.e., one image out of fifty misclassified).

It may seem surprising that we can still detect the CFA interpolation correlations even in the presence of perturbations (e.g., lossy compression, additive noise, non-linear luminance transforms) that may destroy these correlations. The reason for this robustness is that the EM algorithm employs a model of correlations that are already corrupted by noise, Equation (3.42). As a result, the detection accuracy is largely insensitive to small perturbations, and drops gracefully when the amount of perturbation increases.

In summary, we have shown that our detection method can reasonably distinguish between CFA and non-CFA interpolated images even when the images are subjected to lossy compression (JPEG and GIF, but not JPEG2000), additive noise, or luminance non-linearities.

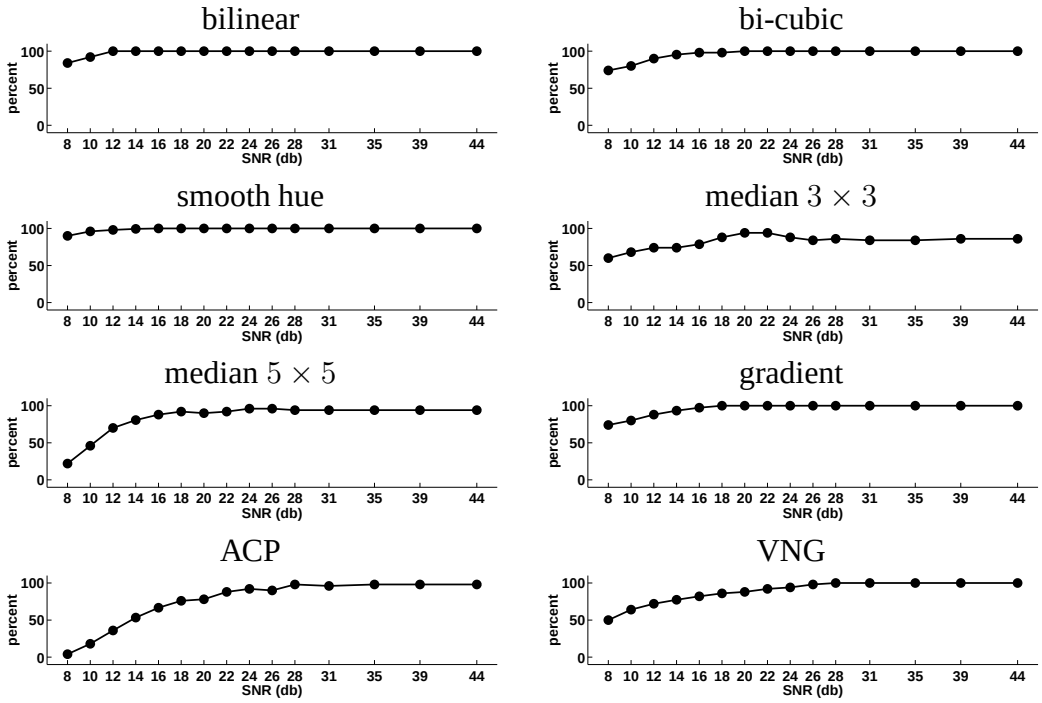

Figure 3.10: Detection accuracies (with 0% false positives) for different CFA interpolation algorithms as a function of signal-to-noise ratio (SNR). Each data point corresponds to the average detection accuracy over fi fty images.

3.4 Summary

Most digital cameras employ a single sensor in conjunction with a color filter array (CFA) — the missing color samples are then interpolated from these recorded samples to obtain a three channel color image. This interpolation introduces specific correlations which are likely to be destroyed when tampering with an image. As such, the presence or lack of correlations produced by CFA interpolation can be used to authenticate an image, or expose it as a forgery. We have shown, for seven different CFA interpolation algorithms, that these correlations are well estimated by a simple linear model. This model, in conjunction with the EM algorithm, was employed to detect manipulated portions of CFA interpolated images. We have shown the efficacy of this approach to detecting traces of digital tampering in lossless and lossy compressed images, and quantified its sensitivity to simple attacks.

In February 2002, Foveon Inc. announced an innovative image sensor that can simultaneously sample the red, green, and blue channels at each pixel location. This CMOS sensor, the Foveon X3, employs three different sensor layers embedded in silicon to exploit the fact that different light frequencies penetrate silicon to different depths. The sensor comes in two versions: 4.5 and 10.2 mega-pixels, and, as of 2004, it has been incorporated in several cameras: the Polaroid x530, the Sigmas SD9 and SD10, and the HanVision HVDUO series. Obviously, digital cameras equipped with the Foveon X3 sensor do not need color filter array interpolation anymore. As a

result, color images from these cameras will no longer exhibit the periodic correlations specific to color filter array interpolated images, and our detection method will not be applicable. We expect, however, that our technique would still prove worthwhile in the future given that single sensor cameras constitute the vast majority of the market today, and are likely to remain cheaper, and more available to average users.

As with any authentication scheme, our approach is vulnerable to counter-attack. A tampered image could, for example, be re-sampled onto a CFA, and then re-interpolated. This attack, however, requires knowledge of the camera's CFA pattern and interpolation algorithm, and is likely to be beyond the reaches of a novice forger.

3.5 CFA Correlations Detection Algorithm

/* Initialize */

choose $\{\alpha_{u,v}^{(0)}\}$ randomly

choose N and σ_0

set p_0 as 1 over the size of the range of possible values of $f(x,y)$

$n = 0$

repeat

/* expectation step */

for each sample location (x,y)

$r(x,y) = \left| f(x,y) - \sum_{u,v=-N}^{N} \alpha_{u,v}^{(n)} f(x+u, y+v) \right|$ /* residual error */

end

for each sample location (x,y)

$P(x,y) = \frac{1}{\sigma_n \sqrt{2\pi}} \exp\left[-r^2(x,y)/2\sigma_n^2\right]$ /* conditional probability */

$w(x,y) = \frac{P(x,y)}{P(x,y)+p_0}$ /* posterior probability */

end

/* maximization step */

compute $\{\alpha_{u,v}^{(n+1)}\}$ as the solution of the linear system in Equation (3.47)

$\sigma_{n+1} = \left(\frac{\sum_{x,y} w(x,y)\, r^2(x,y)}{\sum_{x,y} w(x,y)} \right)^{1/2}$ /* new variance estimate */

$n = n + 1$

until ($\sum_{u,v} \left| \alpha_{u,v}^{(n)} - \alpha_{u,v}^{(n-1)} \right| < \epsilon$) /* stopping condition */

Chapter 4

Double JPEG Compression

Tampering with a digital image generally requires the use of a photo-editing software package, such as *Adobe Photoshop*®. When creating digital forgeries, an image is loaded into the photo-editing software, some manipulations are performed, and the image is re-saved. If JPEG format is used to store the images (a majority of digital cameras have as default setting the option to store images directly in JPEG format) the tampered image has been double JPEG compressed. Double compressed images contain specific artifacts[1] that can be employed to distinguish them from single compressed images. Note, however, that detecting double JPEG compression does not necessarily prove malicious tampering: it is possible, for example, that a user may re-save high quality JPEG images with lower quality to save storage space. The authenticity of a double JPEG compressed image, however, is at least questionable and further analysis would be required.

We start by giving a brief description of the JPEG compression algorithm. Next, we describe the specific artifacts introduced by double JPEG compression and illustrate them on several examples of natural images. Finally, we propose a method for detecting and quantifying double compressed images and analyze its sensitivity on a database of 100 natural images.

4.1 JPEG Compression

JPEG is a standardized image compression procedure proposed by a committee with the same name JPEG (Joint Photographic Experts Committee). To be generally applicable, the JPEG standard [1] specified two compression schemes: a lossless predictive scheme and a lossy scheme based on the discrete cosine transform (DCT). The most popular lossy compression technique is known as the baseline method and encompasses a subset of the DCT-based modes of operation. In this section, a summary of the baseline method will be presented.

The encoding procedure consists of three sequential steps [59], and yields a compressed stream of data:

1. **Discrete Cosine Transform (DCT)**: image samples are grouped into 8×8 blocks in raster

[1]The fact that specific artifacts are present in double JPEG compressed images was also independently noticed in [39].

scan order (left to right, top to bottom) and shifted from unsigned to signed integers (i.e., from range $[0, 255]$ to $[-128, 127]$), then the DCT of the blocks is computed. Let $f(x, y)$ denote an 8×8 image block, then its DCT takes the form:

$$F(\omega_x, \omega_y) = \frac{1}{4} c(\omega_x) c(\omega_y) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x, y) \cos \frac{(2x+1)\omega_x \pi}{16} \cos \frac{(2y+1)\omega_y \pi}{16}, \quad \omega_x, \omega_y = 0, \ldots, 7, \quad (4.1)$$

where $c(\omega) = 1/\sqrt{2}$, for $\omega = 0$, and $c(\omega) = 1$ otherwise.

2. **Quantization**: The DCT coefficients obtained in the previous step are uniformly quantized. The reason for quantization is to represent the DCT coefficients with no greater precision than necessary for a desired visual quality. Quantization is a point-wise operation defined as division by a quantization step followed by rounding to the nearest integer:

$$F_q(\omega_x, \omega_y) = \left\lfloor \frac{F(\omega_x, \omega_y)}{s(\omega_x, \omega_y)} + \frac{1}{2} \right\rfloor, \quad \omega_x, \omega_y = 0, \ldots, 7, \quad (4.2)$$

where $s(\omega_x, \omega_y)$ is a frequency dependent quantization step. The relationship between the JPEG quality, Q, and the quantization steps $s(\omega_x, \omega_y)$ takes the form:

$$s(\omega_x, \omega_y) = \begin{cases} \max\left(\left\lfloor \frac{200-2Q}{100} C(\omega_x, \omega_y) + \frac{1}{2} \right\rfloor, 1 \right) & , 50 \leq Q \leq 100 \\ \left\lfloor \frac{50}{Q} C(\omega_x, \omega_y) + \frac{1}{2} \right\rfloor & , 0 < Q < 50, \end{cases} \quad (4.3)$$

where $C(\omega_x, \omega_y)$ are experimentally determined to be:

$$C(\omega_x, \omega_y) = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}. \quad (4.4)$$

In Equation (4.4), the upper left value (16) corresponds to the DC term, and the bottom right value (99) corresponds to the highest horizontal and vertical frequencies. The upper right value (61) corresponds to the highest vertical frequency, and lowest horizontal frequency, while the bottom left value (72) corresponds to the highest horizontal frequency, and lowest

vertical frequency. These values were determined experimentally to yield a good trade-off between compression and perceptual quality. Note that quantization is a non-invertible operation, therefore causing loss of information. This step represents the main source of information loss in the DCT-based JPEG compression scheme.

3. **Entropy Encoding**: This step involves lossless entropy compression that transforms the quantized DCT coefficients into a stream of compressed data. The most frequently used procedure is Huffman coding, although arithmetic coding is also supported.

The decoding of a compressed data stream involves the inverse of the previous three steps, taken in reverse order:

1. **Entropy Decoding**: The compressed data stream is decoded to recover the quantized DCT coefficients.

2. **De-quantization**: For each 8×8 block, the quantized DCT coefficients are multiplied by their corresponding quantization steps:

$$\tilde{F}(\omega_x, \omega_y) = F_q(\omega_x, \omega_y) s(\omega_x, \omega_y), \quad \omega_x, \omega_y = 0, \ldots, 7, \tag{4.5}$$

   where $s(\omega_x, \omega_y)$ is defined in Equation (4.3).

3. **Inverse Discrete Cosine Transform (IDCT)**: The IDCT of the de-quantized DCT coefficients, $\tilde{F}(\omega_x, \omega_y)$, is first computed, then the result is shifted back to the range $[0, 255]$ to obtain reconstructed 8×8 image blocks. Let $\tilde{f}(x, y)$ denote a reconstructed image block obtained from IDCT:

$$\tilde{f}(x, y) = \frac{1}{4} \sum_{\omega_x=0}^{7} \sum_{\omega_y=0}^{7} c(\omega_x) c(\omega_y) \tilde{F}(\omega_x, \omega_y) \cos \frac{(2x+1)\omega_x \pi}{16} \cos \frac{(2y+1)\omega_y \pi}{16}, \quad x, y = 0, \ldots, 7, \tag{4.6}$$

   where $c(\omega) = 1/\sqrt{2}$, for $\omega = 0$, and $c(\omega) = 1$ otherwise.

For simplicity, only the case of sequential compression of single channel images (gray-scale) has been presented. Three-channel (color) images are first converted to a luminance-chrominance color space [2], the chrominance channels are down-sampled by a factor of two, then the gray-scale compression method is independently applied to the luminance and chrominance channels[2].

Note that double JPEG compression amounts to double quantization of block DCT coefficients. We will show next that the double quantization of a signal of any type (e.g., a sequence of DCT coefficients) introduces specific artifacts that can be quantified and detected.

[2]For the luminance channel the quantization table for gray-scale images, Equation (4.4), is employed. A different quantization table is employed for the chrominance channels.

4.2 Double Quantization

Consider the example of a generic discrete 1-D signal $x[t]$. Quantization is a point-wise operation that is described by a one-parameter family of functions[3] :

$$q_a(u) \quad = \quad \left\lfloor \frac{u}{a} \right\rfloor, \tag{4.7}$$

where a is the quantization step (a strictly positive integer), and u denotes a value in the range of $x[t]$. De-quantization is the inverse operation that brings the quantized values back to their original range: $q_a^{-1}(u) = au$. Note that the functions $q_a(u)$ are not invertible, and that de-quantization is not the inverse function of quantization. Double quantization is also a point-wise operation described by a two-parameter family of functions:

$$q_{ab}(u) \quad = \quad \left\lfloor \left\lfloor \frac{u}{b} \right\rfloor \frac{b}{a} \right\rfloor, \tag{4.8}$$

where a and b are the quantization steps (strictly positive integers). Note that double quantization can be represented as a sequence of three steps: quantization with step b, followed by de-quantization with the same step b, followed by quantization with step a.

Consider an example where the samples of $x[t]$ are normally distributed in the range $[0, 127]$ (e.g., a white noise signal). To illustrate the nature of the double quantization artifacts, the original signal $x[t]$ is quantized in four different ways, and the histograms of the original and the four quantized signals are shown in Figure 4.1. Shown in panel (a) is the histogram of the normally distributed original signal. Shown in panels (b) and (c) are the histograms of single and double quantized signals with steps 2, and 3 followed by 2, respectively. Note that some bins in the histogram of the double quantized signal are empty. This is not surprising since the first quantization places the samples of the original signal into 42 bins, while the second quantization re-distributes them into 64 bins. Note, however, the periodic occurrence of these empty bins. Shown in Figure 4.1, panels (d) and (e), are the histograms of single and double quantized signals with steps 3, and 2 followed by 3, respectively. Note that the even bins contain approximately twice as many samples as their neighboring odd bins. This periodic artifact would also be expected, since the even bins receive samples from four original histogram bins, while the odd bins receive samples from only two. Note, however, that in both cases the single quantized signals do not exhibit any periodic artifacts.

To better understand why the double quantization of a signal introduces periodic artifacts in its histogram, we will analyze the dependence between the histograms of original signals, and single and double quantized signals. Consider first the case of a single quantized signal denoted by $x_a[t]$, and denote the histograms of the original and quantized signals by $h(u)$ and $h_a(v)$, where $h(u)$, for example, represents the number of samples of $x[t]$ that take the value u. The relationship between the original and quantized signals takes the form: $x_a[t] = q_a(x[t])$. Since $q_a(\cdot)$ is a many-to-one function several values from the range of $x[t]$ will map onto the same value in the range of $x_a[t]$,

[3]For the purpose of illustration and in order to make the analysis easier we will use the fbor function in the quantization function. Similar results can be shown if integer rounding is used instead.

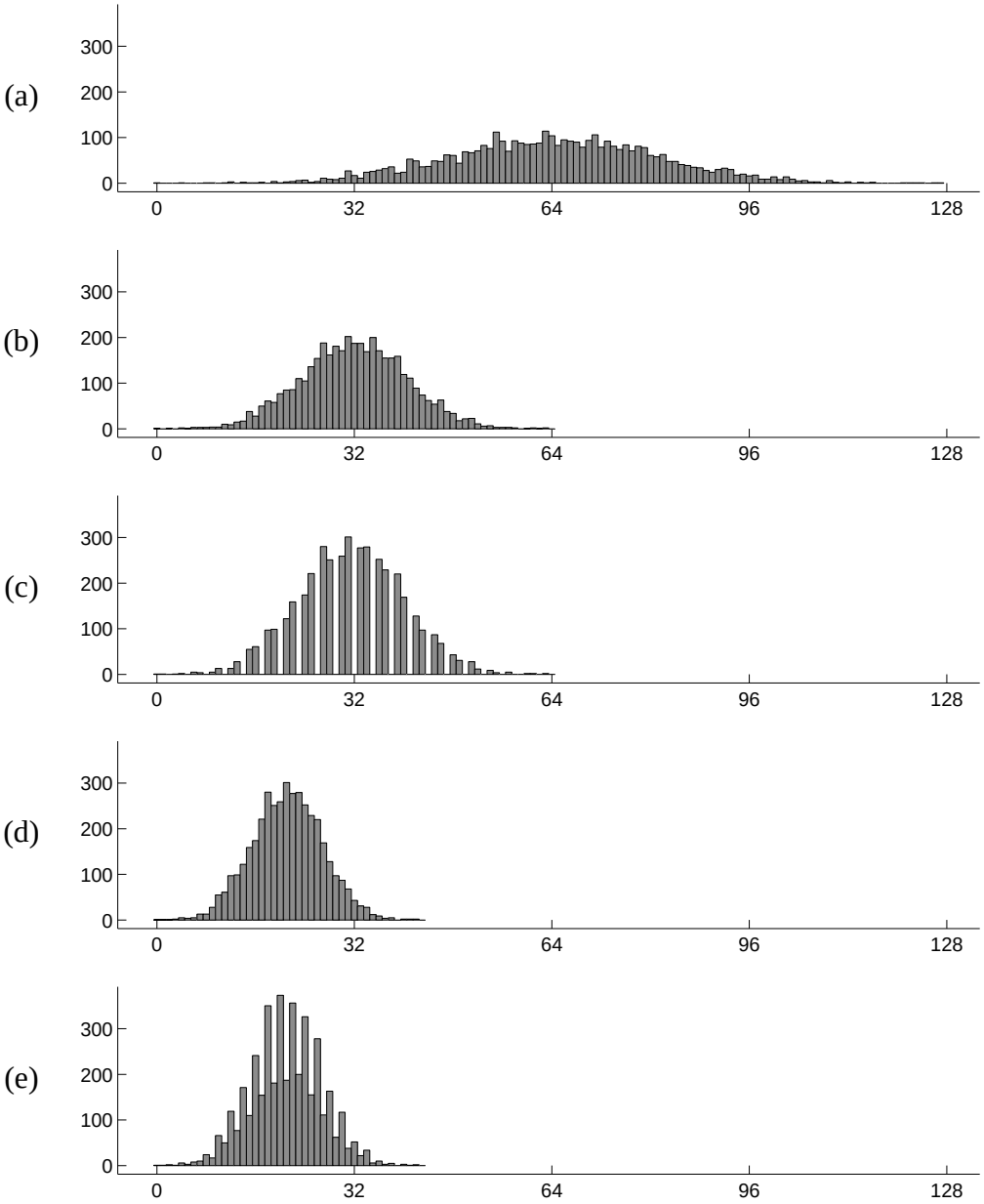

Figure 4.1: Shown are: (a) the histogram of a normally distributed signal; (b), (d) the histograms of single quantized signals with quantization steps 2, and 3; (c), (e) the histograms of double quantized signals with quantization steps 3 followed by 2, and 2 followed by 3. Note the presence of periodic artifacts in the histograms of double quantized signals. Note also that such artifacts are not present in the histograms of single quantized signals.

i.e., several bins from h contribute to a bin in h_a. For example, let v denote a value in the range of $x_a[t]$, then the values in the range of $x[t]$ that map to it are in the interval $[av, av + (a-1)]$. Therefore, the relationship between $h(u)$ and $h_a(v)$ is given by:

$$h_a(v) \;=\; \sum_{k=0}^{a-1} h(av+k). \tag{4.9}$$

Note that there are exactly a bins in the original histogram that contribute to each bin in the histogram of the quantized signal. Consider next the case of a double quantized signal, denoted by $x_{ab}[t]$, and let its histogram be denoted by $h_{ab}(v)$. The relationship between the double quantized and original signals is given by: $x_{ab}[t] = q_{ab}(x[t])$. In contrast to the single quantization case, the number of bins of h that contribute to a bin of h_{ab} will depend on the double quantized bin value. Let v be a value in the range of $x_{ab}[t]$. Denote by u_{min} and u_{max} the smallest and largest values, respectively, in the range of $x[t]$ that map to v, i.e., satisfy the equation:

$$\left\lfloor \left\lfloor \frac{u}{b} \right\rfloor \frac{b}{a} \right\rfloor \;=\; v. \tag{4.10}$$

Using the following property of the floor function:

$$\lfloor z \rfloor = m \;\;\Rightarrow\;\; m \le z < m+1, \tag{4.11}$$

where z is an arbitrary real and m is the largest integer smaller than z, Equation (4.10) yields:

$$v \le \left\lfloor \frac{u}{b} \right\rfloor \frac{b}{a} < v+1 \quad \Leftrightarrow \quad \frac{a}{b}v \le \left\lfloor \frac{u}{b} \right\rfloor < \frac{a}{b}(v+1). \tag{4.12}$$

Since $\lfloor u/b \rfloor$ is an integer , Equation (4.12) can be rewritten using the ceiling function to include only integers:

$$\left\lceil \frac{a}{b}v \right\rceil \le \left\lfloor \frac{u}{b} \right\rfloor \le \left\lceil \frac{a}{b}(v+1) \right\rceil - 1. \tag{4.13}$$

From Equation (4.13) it can be seen that u_{min} must satisfy:

$$\left\lfloor \frac{u_{min}}{b} \right\rfloor = \left\lceil \frac{a}{b}v \right\rceil \quad \Leftrightarrow \quad u_{min} = \left\lceil \frac{a}{b}v \right\rceil b, \tag{4.14}$$

while u_{max} must satisfy:

$$\left\lfloor \frac{u_{max}}{b} \right\rfloor = \left\lceil \frac{a}{b}(v+1) \right\rceil - 1 \quad \Leftrightarrow$$
$$u_{max} = \left(\left\lceil \frac{a}{b}(v+1) \right\rceil - 1 \right) b + (b-1) = \left\lceil \frac{a}{b}(v+1) \right\rceil b - 1. \tag{4.15}$$

Since double quantization is a monotonically increasing function, it follows that all the values between u_{min} and u_{max} will map to v through double quantization. The relationship between the original and double quantized histograms takes the form:

$$h_{ab}(v) \;=\; \sum_{u=u_{min}}^{u_{max}} h(u). \tag{4.16}$$

Note that the number of original histogram bins contributing to bin v in the double quantized histogram depends on v, and let $n(v)$ denote this number. Using Equation (4.14) and Equation (4.15) the value of $n(v)$ is given by[4]:

$$n(v) = u_{max} - u_{min} + 1 = b\left(\left\lceil \frac{a}{b}(v+1)\right\rceil - \left\lceil \frac{a}{b}v\right\rceil\right). \tag{4.17}$$

Note that $n(v)$ is a periodic function with period b, i.e., $n(v) = n(v + kb)$, where k is any integer. This periodicity is the reason that periodic artifacts appear in histograms of double quantized signals.

Using Equation (4.17), the double quantization artifacts shown in Figure 4.1 can be explained. Consider first the case of double quantization using steps $b = 3$ followed by $a = 2$, Figure 4.1(c). The number of original histogram bins contributing to double quantized histogram bins of the form $(3k + 2)$, with k integer, is given by:

$$\begin{aligned} n(3k+2) &= 3\left(\left\lceil \frac{2}{3}(3k+3)\right\rceil - \left\lceil \frac{2}{3}(3k+2)\right\rceil\right) \\ &= 3\left(\lceil 2k+2\rceil - \left\lceil 2k+\frac{4}{3}\right\rceil\right) \\ &= 3\left(2k+2-2k-\left\lceil \frac{4}{3}\right\rceil\right) = 0. \end{aligned} \tag{4.18}$$

This is consistent with the observation that every $(3k+2)^{nd}$ (k integer) bin of the double quantized histogram is empty. In the other example of double quantization, when steps $b = 2$ followed by $a = 3$ are employed in Equation (4.17), it can be shown that $n(2k) = 4$ and $n(2k + 1) = 2$, with k integer. Again this is consistent with Figure 4.1(e).

There are cases when the histogram of a double quantized signal does not contain periodic artifacts. For example, if in Equation (4.17) a/b is an integer then $n(v) = a$. Note that the same result is obtained if the signal were single quantized with step a. In this case, single and double quantized signals have the same histogram, therefore it is impossible to distinguish between them. Note also in Equation (4.16) that the histogram of the double quantized signal, h_{ab}, depends on the values of the histogram of the original signal h. It is conceivable that histograms of original signals may contain naturally occurring artifacts that could mask those introduced by double quantization. While this may happen on occasion, such artifacts do not occur often. For example, the histograms of DCT coefficients of natural images are typically well behaved, and double quantization introduces detectable periodic artifacts in most cases.

4.3 Results

Shown in Figure 4.2 are examples of gray-scale and color images that have been JPEG compressed. Shown in the left column are images single compressed with qualities 75 (Figure 4.2(a) and (c)),

[4]If instead of the fbor function, integer rounding is employed for quantization, it can be shown that $n(v)$ takes the form: $n(v) = b\left(\left\lceil \frac{a}{b}(v+\frac{1}{2})\right\rceil - \left\lceil \frac{a}{b}(v-\frac{1}{2})\right\rceil\right)$.

and 85 (Figure 4.2(b) and (d)), and in the middle column are the same images images double compressed with qualities 85 followed by 75 (Figure 4.2(a) and (c)), and 75 followed by 85 (Figure 4.2(b) and (d)). Also shown in the right column are the absolute differences between the single and double compressed images[5]. Note that although there is little perceptual difference between the single and double compressed images, double compression introduces band-pass noise with a signal-to-noise ratio ranging between 20db and 24dB. Since double JPEG compression amounts to double quantization of DCT coefficients of image blocks, the histograms of these coefficients will contain periodic artifacts, as explained in Section 4.2. For example, shown in Figure 4.3 are the DCT coefficients, their histograms and the Fourier transforms of the zero-mean histograms of the gray-scale images in Figure 4.2(a). Shown in Figure 4.3(a)-(b) are the DCT coefficients of the single JPEG compressed image with quality 75, while in Figure 4.3(c)-(d) are the DCT coefficients of the double JPEG compressed image with qualities 85 followed by 75. The DCT coefficients are shown as images (auto-scaled to fill the full intensity range), where each pixel corresponds to a 8×8 block in the JPEG compressed images, and the intensities represent the coefficient values. The coefficients in Figure 4.3(a) and (c) correspond to DCT frequency $(1, 1)$, the DC component, and the coefficients in Figure 4.3(b) and (d) correspond to DCT frequency $(2, 2)$. In a way similar to Figure 4.3, Figures 4.4-4.6 contain the DCT coefficients[6], their histograms and the Fourier transforms of the zero-mean histograms of the images in Figure 4.2(b)-(d). Note that periodic artifacts are present in the histograms of the DCT coefficients of the double compressed images. These artifacts are particularly visible in the Fourier transforms of the histograms in the form of strong peaks in the mid and high frequencies. Note that for single compressed images no periodic artifacts or high-frequency peaks are present.

The periodic patterns introduced by double JPEG compression depend on the quality parameters. As a result, it is possible to detect not only if an image has been double compressed, but also the compression qualities that have been used. The second parameter can be found from the quantization table stored in the JPEG file, while the first parameter can be inferred from the location of the frequency peaks in the Fourier transform of the DCT coefficient histograms.

4.3.1 Quantifying Double JPEG Compression Artifacts

For forensics applications, it is important to quantify the sensitivity and robustness of our approach for detecting double JPEG compression. To this end it is first necessary to devise a quantitative measure for the periodic artifacts introduced in the DCT coefficient histograms. The purpose of this measure is to discriminate between single and double quantized coefficients, e.g., high values would correspond to double quantization, while low values would correspond to single quantization.

Consider a sequence of DCT coefficients known to have been double quantized with steps b followed by a. If a/b does not have an integer value, Section 4.2, double quantization introduces in the DCT coefficient histogram periodic artifacts that have a specific peak pattern in the frequency

[5]For display purposes, these absolute differences have been auto-scaled to fi ll the full intensity range, and gamma corrected with $\gamma = 0.75$.

[6]For color images the DCT coeffi cients of the luminance channel are shown.

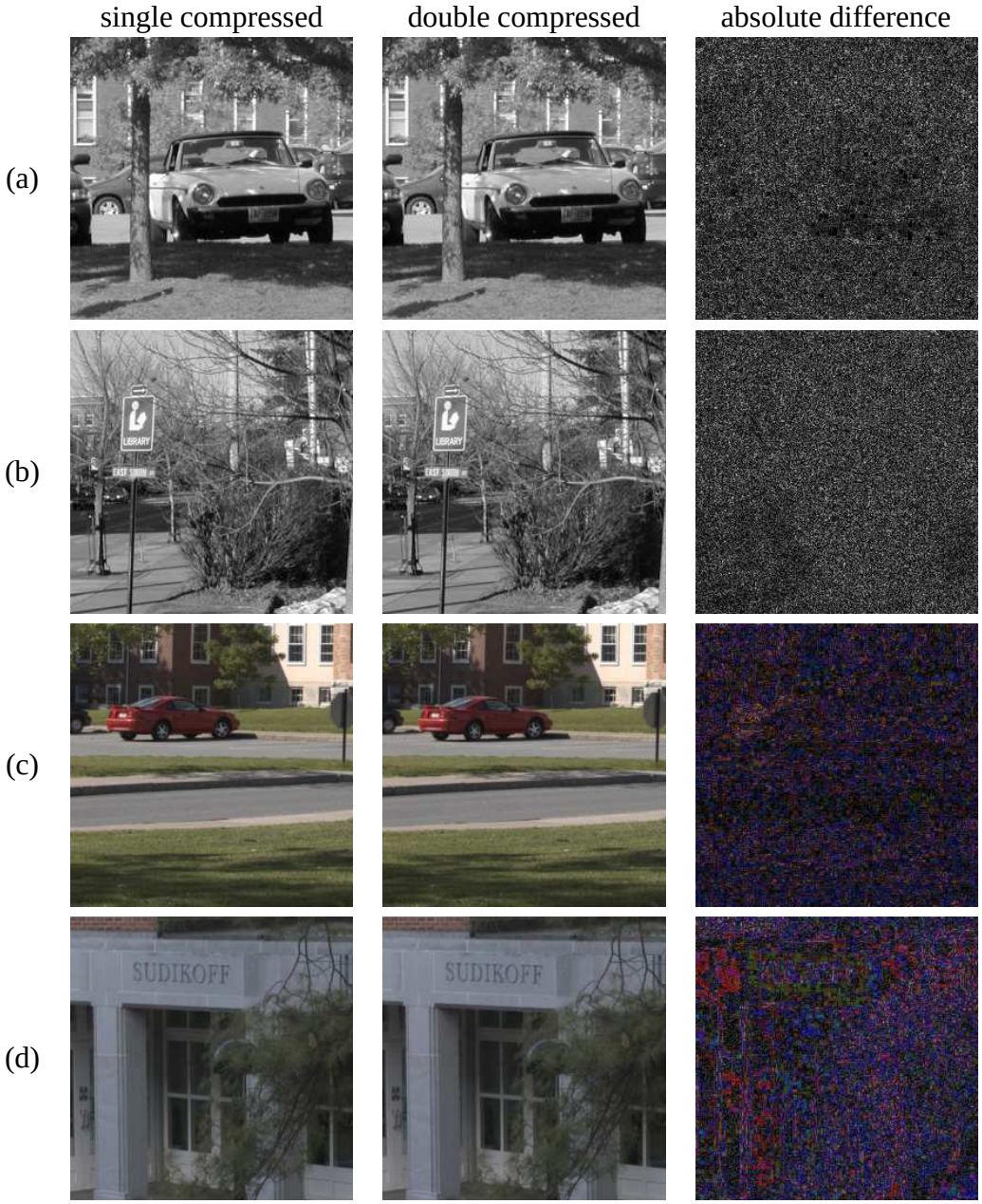

Figure 4.2: Shown on the left are four images single JPEG compressed with qualities 75, (a) and (c), and 85, (b) and (d). Shown in the middle are the same images double JPEG compressed with qualities 85 followed by 75, (a) and (c), and 75 followed by 85, (b) and (d). Shown on the right are the absolute differences of the single and double JPEG compressed images (for display purposes, these differences were gamma corrected with $\gamma = 0.75$, and auto-scaled to fill the full intensity range). See also Figures 4.3-4.6.

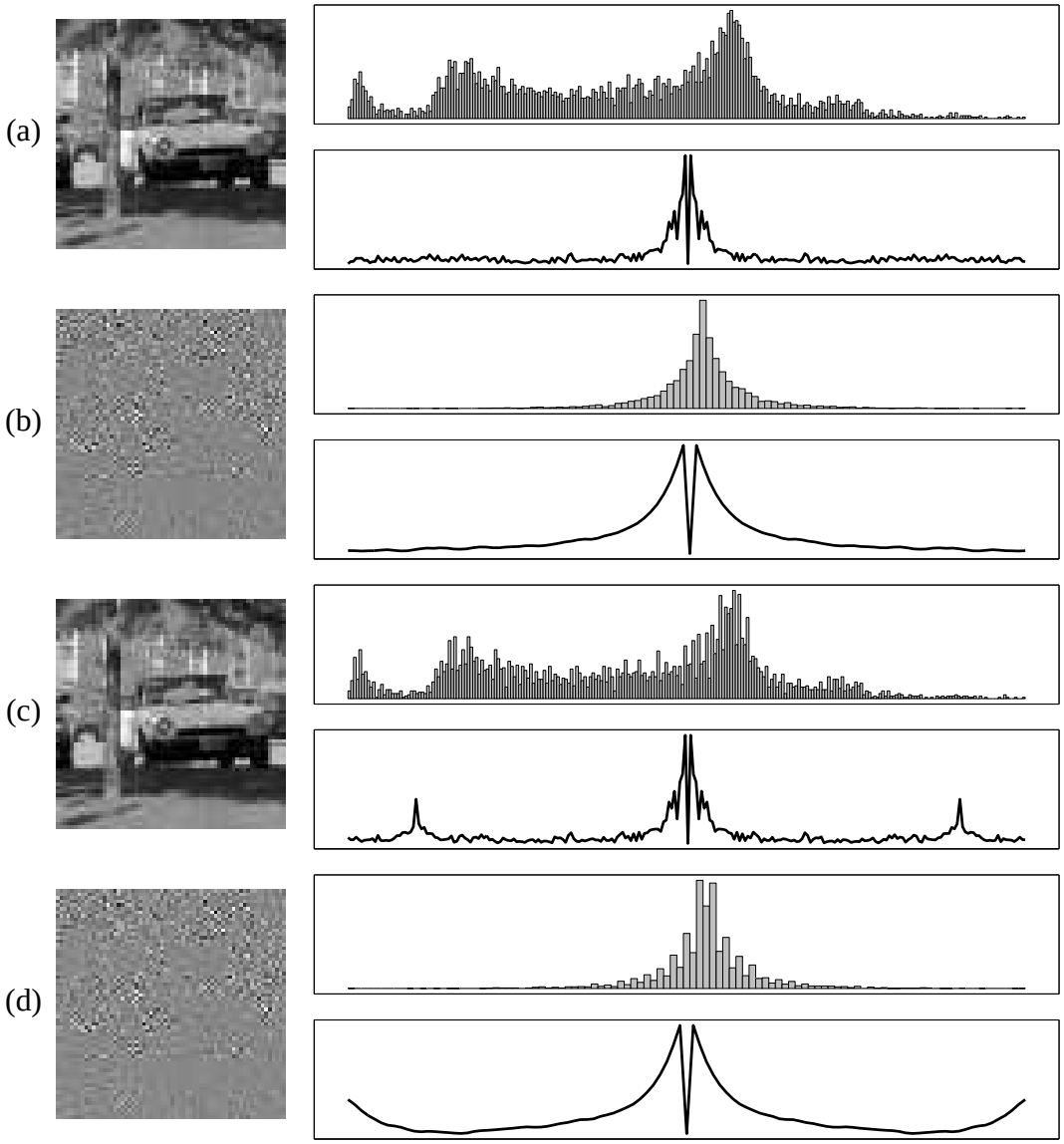

Figure 4.3: Shown are DCT coeffi cients, their histograms and the Fourier transform of the zero-mean histograms for a gray-scale image single JPEG compressed with quality 75, (a) and (b), and double JPEG compressed with qualities 85 followed by 75, (c) and (d). The DCT coeffi cients in panels (a) and (c) correspond to DCT frequency (1,1), the DC component, and those in panels (b) and (d) correspond to DCT frequency (2,2). See also Figure 4.2.

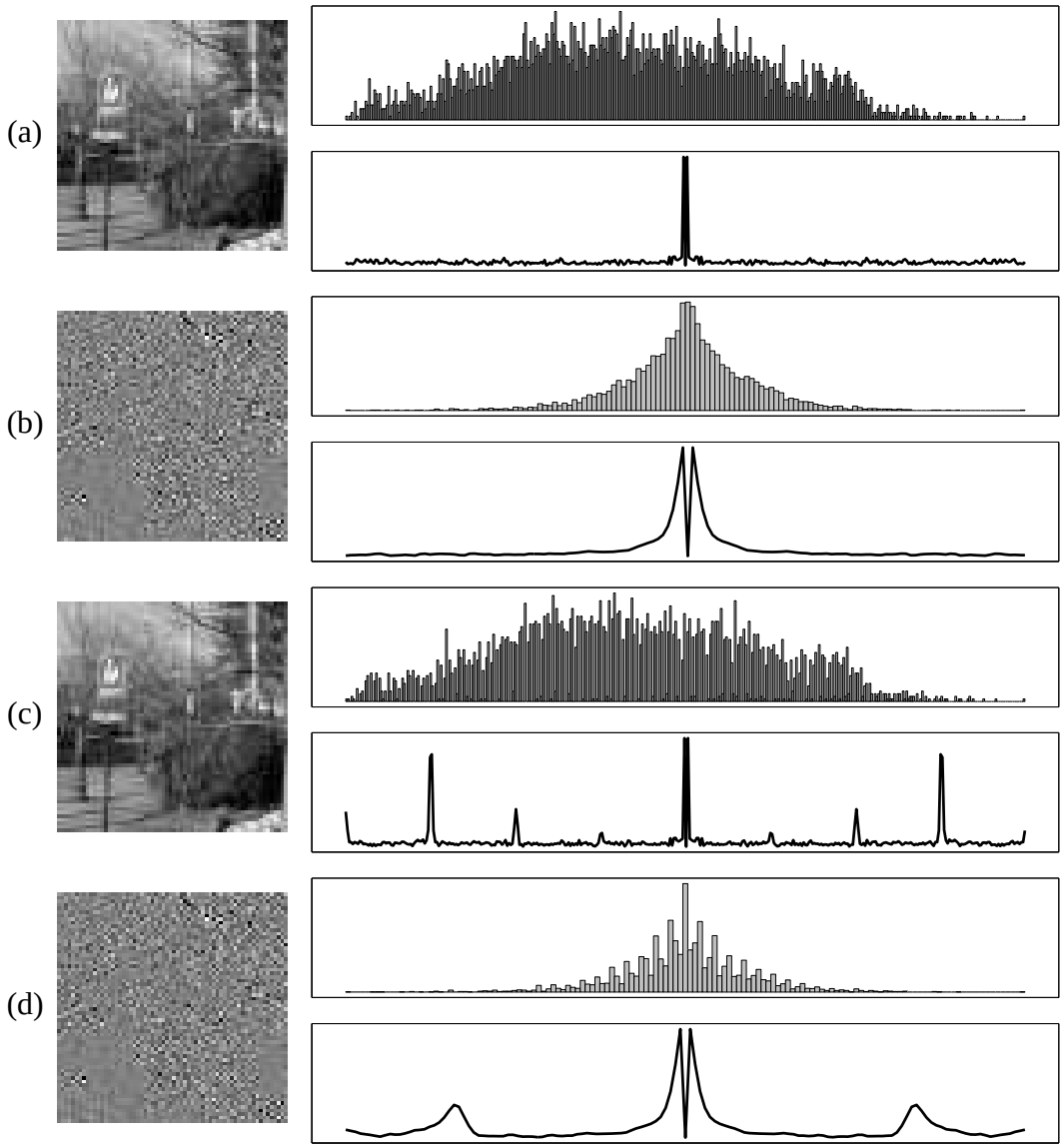

Figure 4.4: Shown are the DCT coeffi cients, their histograms and the Fourier transform of the zero-mean histograms for a gray-scale image single JPEG compressed with quality 85, (a) and (b), and double JPEG compressed with qualities 75 followed by 85, (c) and (d). The DCT coeffi cients in panels (a) and (c) correspond to DCT frequency (1,1), the DC component, and those in panels (b) and (d) correspond to DCT frequency (2,2). See also Figure 4.2.

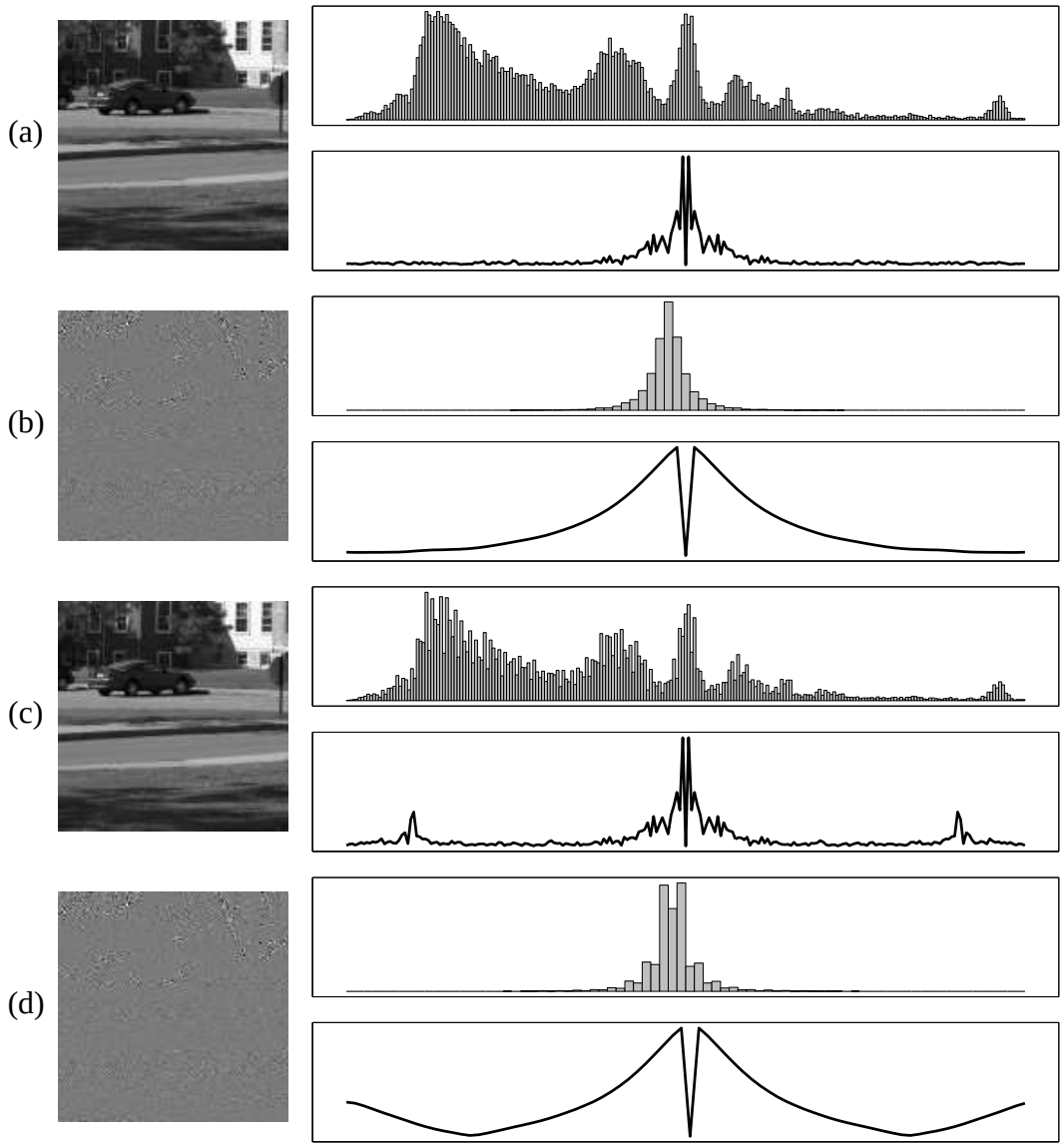

Figure 4.5: Shown are DCT coeffi cients, their histograms and the Fourier transform of the zero-mean histograms of the luminance channel of a color image, single JPEG compressed image with quality 75, (a) and (b), and double JPEG compressed image with qualities 85 followed by 75, (c) and (d). The DCT coeffi cients in panels (a) and (c) correspond to the DCT frequency (1,1), the DC component, and those in panels (b) and (d) correspond to the DCT frequency (2,2). See also Figure 4.2.

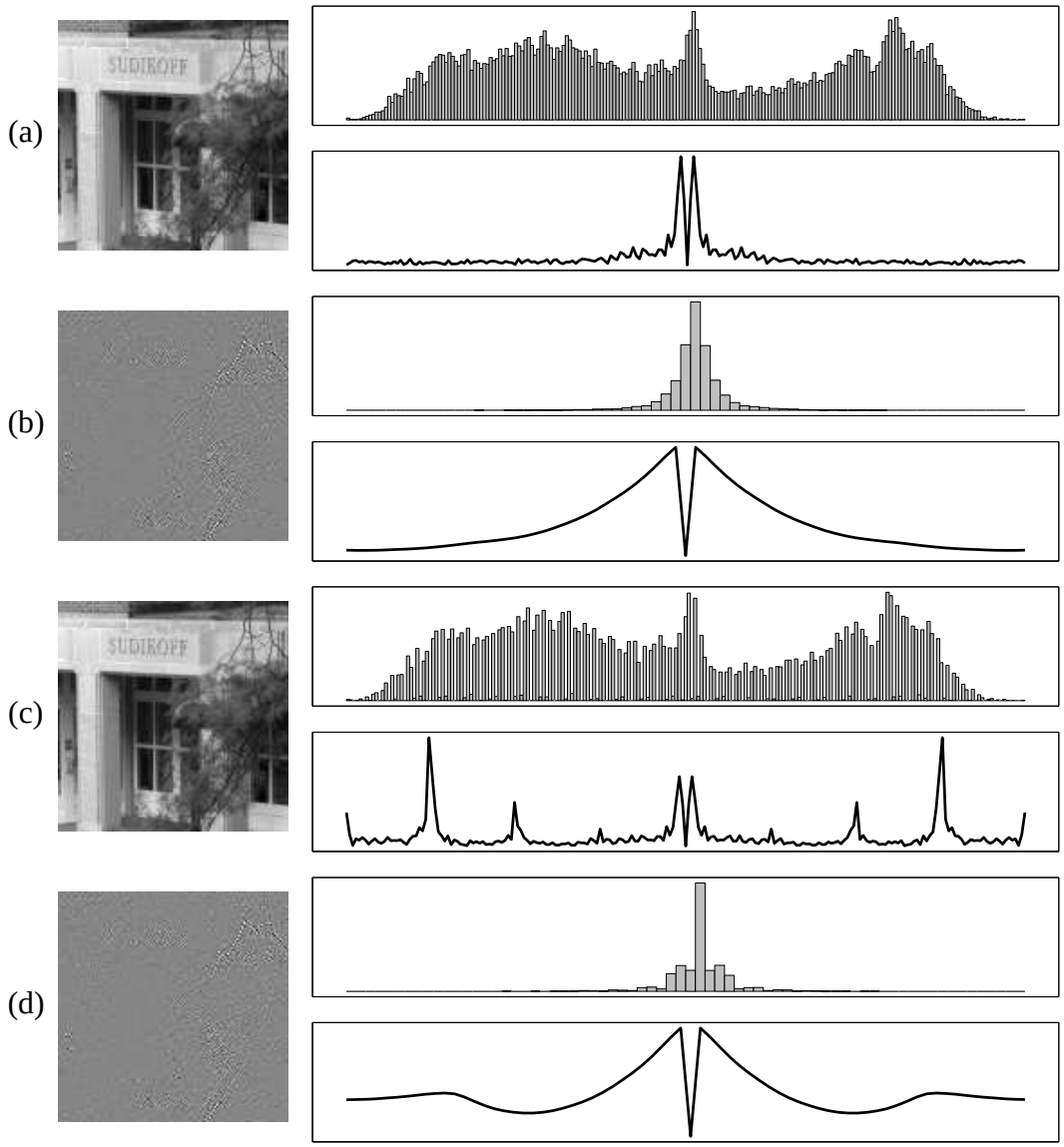

Figure 4.6: Shown are the DCT coefficients, their histograms and the Fourier transform of the zero-mean histograms of the luminance channel of a color image, single JPEG compressed image with quality 85, (a) and (b), and double JPEG compressed image with qualities 75 followed by 85, (c) and (d). The DCT coefficients in panels (a) and (c) correspond to the DCT frequency (1,1), the DC component, and those in panels (b) and (d) correspond to the DCT frequency (2,2). See also Figure 4.2.

domain. This pattern can be estimated by taking a signal drawn from a uniform distribution, double quantizing it with the same steps (b followed by a), computing the Fourier transform of the double quantized signal's histogram, and finding the peak locations. A quantitative measure for the periodicity of the DCT coefficients' histogram is then obtained by averaging the energy values in the frequency domain at the peak locations. More specifically, denoting with $h_{ab}(v)$ and $u_{ab}(v)$ the histograms of the DCT coefficients and of the double quantized, uniformly distributed signal, respectively, the quantitative measure is computed as follows:

1. The histograms h_{ab} and u_{ab} are Fourier transformed:

$$H_{ab}(\omega) = h_{ab}(v) \qquad\qquad U_{ab}(\omega) = u_{ab}(v). \tag{4.19}$$

2. The double quantization peaks are located in $U_{ab}(\omega)$ by selecting the values above a threshold. Let P denote the set of peak locations in the frequency domain:

$$P = \{\omega \mid U_{ab}(\omega) \geq E_t\}, \tag{4.20}$$

   where E_t is an empirically chosen threshold[7].

3. The Fourier transform of the DCT coefficient histogram typically contains a decaying trend that interferes with the peaks introduced by double quantization. This trend needs to be removed in order to obtain an accurate measurement of the peaks' energy. We have empirically found that a two-parameter generalized Laplace model approximates well, in general, the shape of this trend[8]:

$$L(\omega; \alpha, \beta) = e^{-|\omega|^{\alpha}/\beta}. \tag{4.21}$$

   The model parameters are estimated through non-linear minimization to be optimal in the least squares sense: $(\alpha^{\star}, \beta^{\star}) = \arg\min_{(\alpha,\beta)} E(\alpha, \beta)$, where the quadratic error function $E(\alpha, \beta)$ takes the form:

$$E(\alpha, \beta) = \sum_{\omega \notin P} \left(H_{ab}(\omega) - L(\omega; \alpha, \beta)\right)^2. \tag{4.22}$$

   Note that, in order to avoid the interference of the double quantization peaks, only the values at non-peak locations, $\omega \notin P$, are employed in the computation of the error function.

   The decaying trend is removed by subtracting the generalized Laplace model, $L(\omega; \alpha^{\star}, \beta^{\star})$, from $H_{ab}(\omega)$ and taking the square to avoid negative values:

$$D_{ab}(\omega) = (H_{ab}(\omega) - L(\omega; \alpha^{\star}, \beta^{\star}))^2. \tag{4.23}$$

[7] For our experiments we have chosen E_t to be the 85^{th} percentile of the values of $U_{ab}(\omega)$.

[8] The generalized Laplace model, also known as generalized Gaussian or stretched exponential, has been used in the past to model the marginal densities of wavelet coeffi cients [42, 57, 45]. The exact form of the generalized Laplace distribution is given by $\frac{e^{-|x/s|^p}}{Z(s,p)}$, where the normalizing factor is $Z(s,p) = 2\frac{s}{p}\Gamma\left(\frac{1}{p}\right)$, with the generalized factorial gamma function given by: $\Gamma(z) = \int_0^{\infty} t^{z-1}e^{-t}dt$.

4. From $D_{ab}(\omega)$ the periodicity measure is computed as a weighted average at the peak locations in P:

$$M(h_{ab}; a, b) = \frac{\sum_{\omega} W(\omega) D_{ab}(\omega)}{\sum_{\omega} W(\omega)}, \tag{4.24}$$

where the weights $W(\omega)$ are given by:

$$W(\omega) = \begin{cases} U_{ab}(\omega) & \omega \in P \\ 0 & \omega \notin P. \end{cases} \tag{4.25}$$

Note that the measure in Equation (4.24) depends on the histogram of the DCT coefficients, h_{ab} as well as on the quantization steps, a and b. Since in general the first quantization step, b, is unknown, a measure that is independent of b is obtained by taking the maximum over all detectable values of b:

$$M(h_{ab}; a) = \max_{b \in B_a} M(h_{ab}; a, b), \tag{4.26}$$

where the set of all detectable first steps is given by: $B_a = \{b \mid a/b \notin \mathbb{N}, b \leq b_{\text{max}}\}$, where b_{max} is a positive integer.

Consider a sequence of DCT coefficients known to be quantized with step a and let $h(v)$ denote its histogram. If the value of the measure, $M(h; a)$, is above a specified threshold, it is assumed that double quantization periodic artifacts are present in the DCT coefficient histogram and the DCT coefficients are labeled as double quantized. The threshold is empirically determined from single quantized DCT coefficients so that a desired false positive rate is obtained.

Shown in Figures 4.7-4.8, panels (a)-(d), are the histograms and their Fourier transforms for a sequence of DCT coefficients, and for a signal drawn from a uniform distribution that have been both double quantized with steps 3 followed by 7, and 4 followed by 7, respectively. The DCT coefficients correspond to DCT frequencies $(2, 1)$ and $(1, 2)$, and have been taken from the luminance channel of images double JPEG compressed with qualities 85, followed by 70. Note that the high frequency peaks appear at the same locations in the Fourier transforms of both histograms. Note also that the spectrum of the double quantized uniform distribution histogram is flat except at the locations of the peaks, while the spectrum of the DCT coefficient histogram contains a decaying trend. Shown in Figures 4.7-4.8, panels (e)-(f), are the generalized Laplace models of the DCT coefficient spectra and the square differences between the spectra and the models. The values of the optimal (in the least squares sense) model parameters were: $\alpha = 0.94961$, $\beta = 6.3683$ for Figure 4.7(e), and $\alpha = 1.2316$, $\beta = 5.3878$ for Figure 4.8(e). Note the similarity between the spectra of the double quantized uniform distributions, on one hand, and the difference between the DCT coefficient spectra and their models, on the other hand. The values used to compute the weighted average in Equation (4.24) are indicated by the bigger white dots in panels (d) and (f) of Figures 4.7-4.8.

4.3.2 Sensitivity and Robustness

With the ability to quantitatively measure the double quantization periodic artifacts, we have devised a simple scheme to detect double JPEG compressed images. Given a JPEG compressed

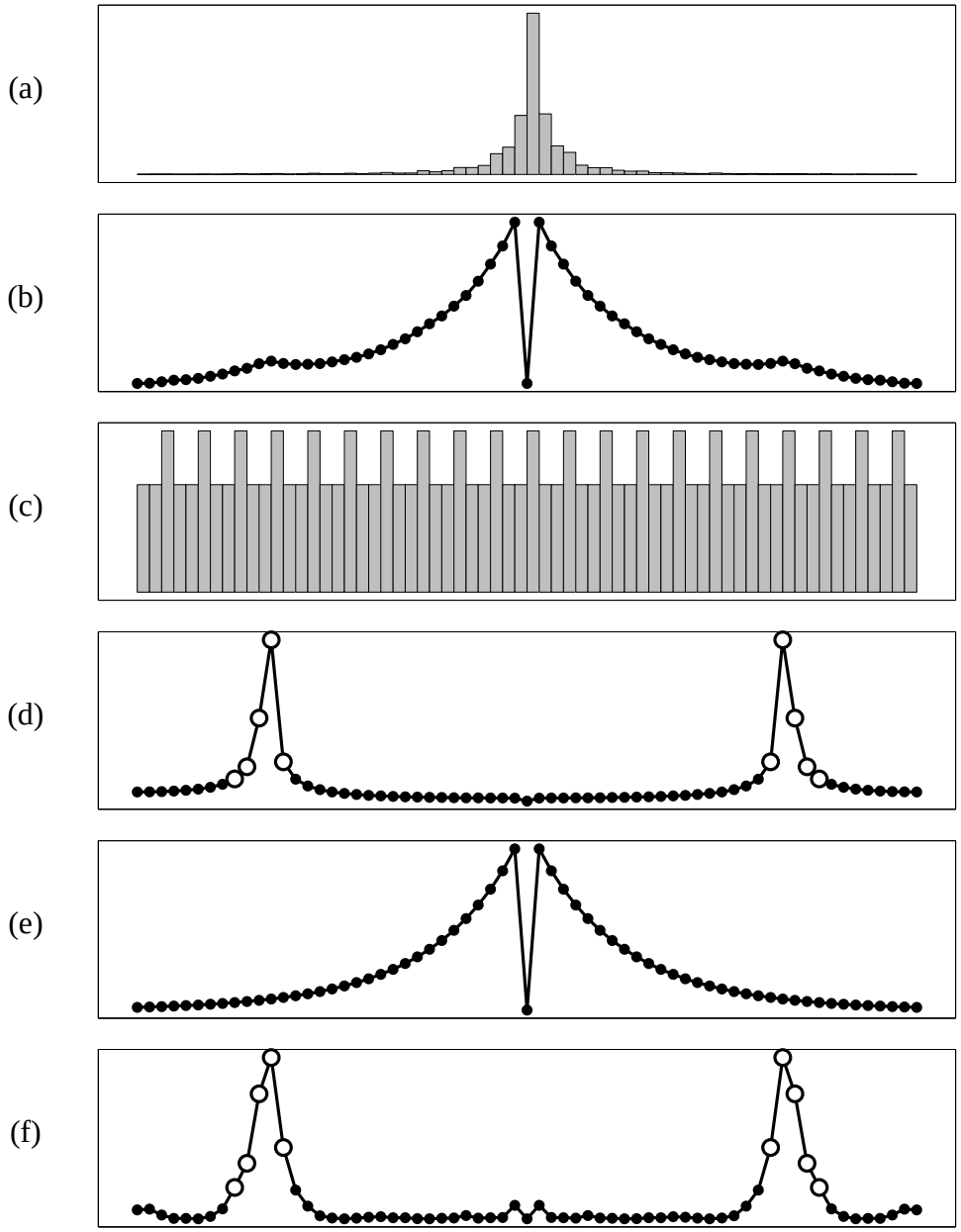

Figure 4.7: Shown are: (a) the histogram of the DCT coeffi cients, corresponding to DCT frequency $(2, 1)$, of a double JPEG compressed image with qualities 85 followed by 70 (the quantization steps were 3, and 7, respectively); (b) the Fourier transform of the zero-mean histogram in panel (a); (c) the histogram of a signal drawn from a uniform distribution, double quantized with steps 3 followed by 7; (d) the Fourier transform of the zero-mean histogram in panel (c); (e) the generalized Laplace model that best approximates (in the least squares sense) the Fourier transform in panel (b); (f) the square difference between the Fourier transform and the model, panels (b) and (e).

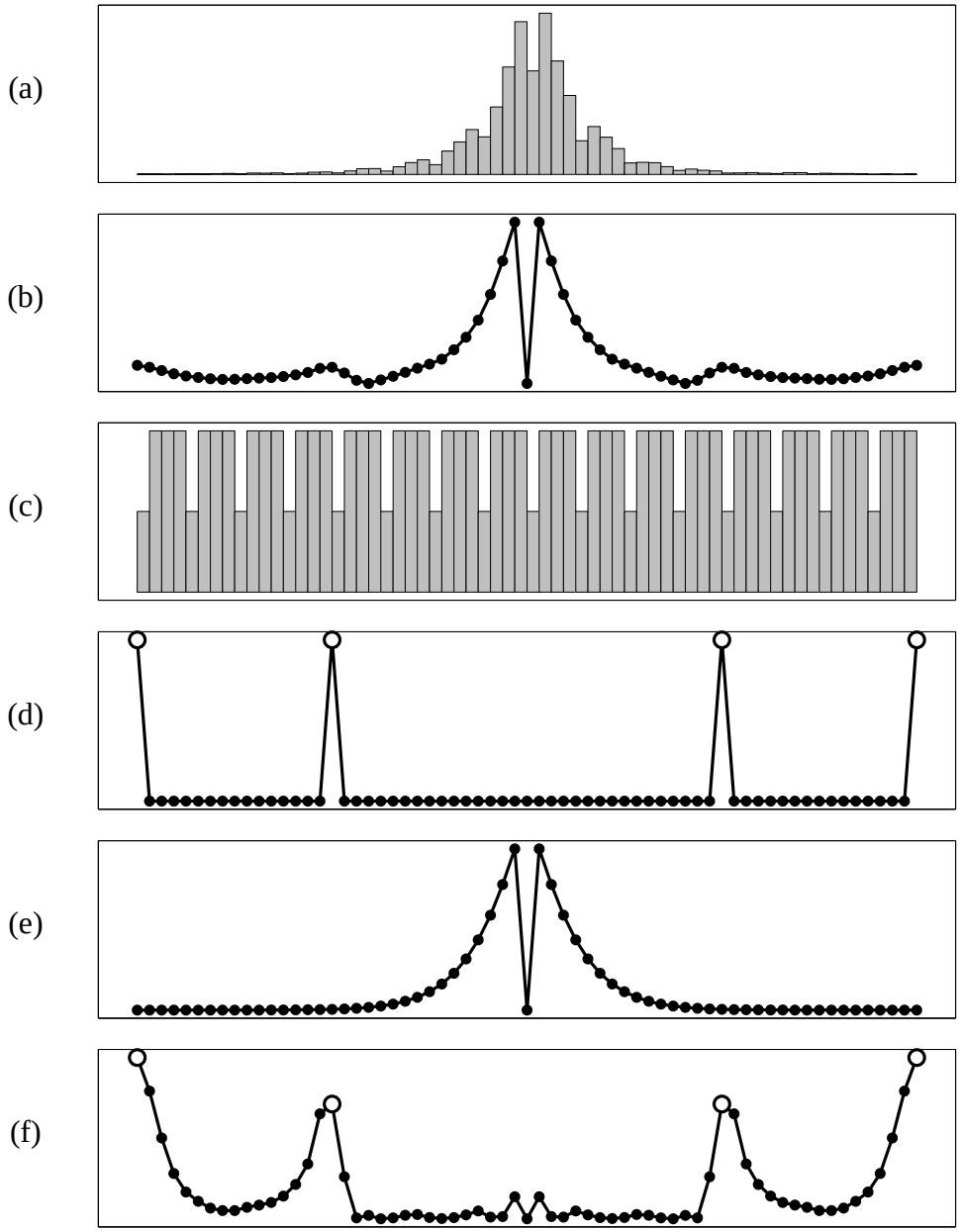

Figure 4.8: Shown are: (a) the histogram of the DCT coeffi cients, corresponding to DCT frequency $(1, 2)$, of a double JPEG compressed image with qualities 85 followed by 70 (the quantization steps were 4, and 7, respectively); (b) the Fourier transform of the zero-mean histogram in panel (a); (c) the histogram of a signal drawn from a uniform distribution, double quantized with steps 4 followed by 7; (d) the Fourier transform of the zero-mean histogram in panel (c); (e) the generalized Laplace model that best approximates (in the least squares sense) the Fourier transform in panel (b); (f) the square difference between the Fourier transform and the model, panels (b) and (e).

Q_1 \ Q_2	50	55	60	65	70	75	80	85	90	95
50	–	100%	100%	100%	100%	100%	100%	100%	100%	100%
55	70%	–	99%	100%	100%	100%	100%	100%	100%	100%
60	98%	85%	–	100%	100%	100%	100%	100%	100%	100%
65	100%	100%	100%	–	100%	100%	100%	100%	100%	100%
70	100%	100%	100%	97%	–	100%	100%	100%	100%	100%
75	93%	100%	100%	100%	100%	–	100%	100%	100%	100%
80	96%	100%	98%	100%	100%	100%	–	100%	100%	100%
85	74%	65%	100%	98%	100%	100%	100%	–	100%	100%
90	48%	63%	78%	89%	99%	100%	100%	100%	–	100%
95	N/D	N/D	73%	98%	N/D	N/D	N/D	100%	100%	–

Table 4.1: Detection accuracy with 0% false positive rate for double JPEG compressed images. The first quality is held constant on the rows, while the second quality is held constant on the columns. The N/D (not detectable) table entries correspond to pairs of qualities that yield double compressed images indistinguishable from single compressed images. Each accuracy value corresponds to an average over 100 images. See also Figure 4.9.

image, we first extract the coefficients of a subset of DCT frequencies, and the quantization table stored in the JPEG file. Next, we employ the periodicity measure defined in the previous section, Equation (4.26), to check if there are double quantized coefficients. If the coefficients of at least one DCT frequency are double quantized, the image is labeled as double JPEG compressed. A more detailed description of this method is presented in Section 4.5.

To test our detection method we have built a database of 100 color images, 1024×1024 in size. These images were cropped from approximately 35 larger 2000×3008 images taken with a Nikon D100 camera and stored in RAW format. Each image was double JPEG compressed with several pairs of qualities. The two chrominance channels of a JPEG compressed image are down-sampled by a factor of two, and quantized using large quantization steps. As a result, the histograms of the chrominance DCT coefficients contain little information, which makes the detection of periodic double quantization artifacts difficult. For our detection experiments, we have employed only the histograms of the DCT coefficients from the luminance channel. The parameters of our detection algorithm were fixed throughout to be: $N = 10$, $V_1 = \{-128, \ldots, 127\}$, $V_2, V_3 = \{-64, \ldots, 63\}$, $V_4, \ldots, V_{10} = \{-32, \ldots, 31\}$,

Shown in Table 4.1 and Figure 4.9 are the detection accuracies of our method when applied to single and double JPEG compressed database images. The false positive rate was 0%, i.e., no single compressed image is misclassified. Each table value and graph data point corresponds to an average over the 100 images. In general, the detection accuracies are nearly perfect. Note that in Table 4.1 several entries whose first quality is 95 are marked N/D (not detectable), e.g., $Q_1 = 95$ and $Q_2 = 80$. This is because, for the DCT frequencies considered, the ratios between the second and first quantization steps have integer values, in which case, as explained in Section 4.2,

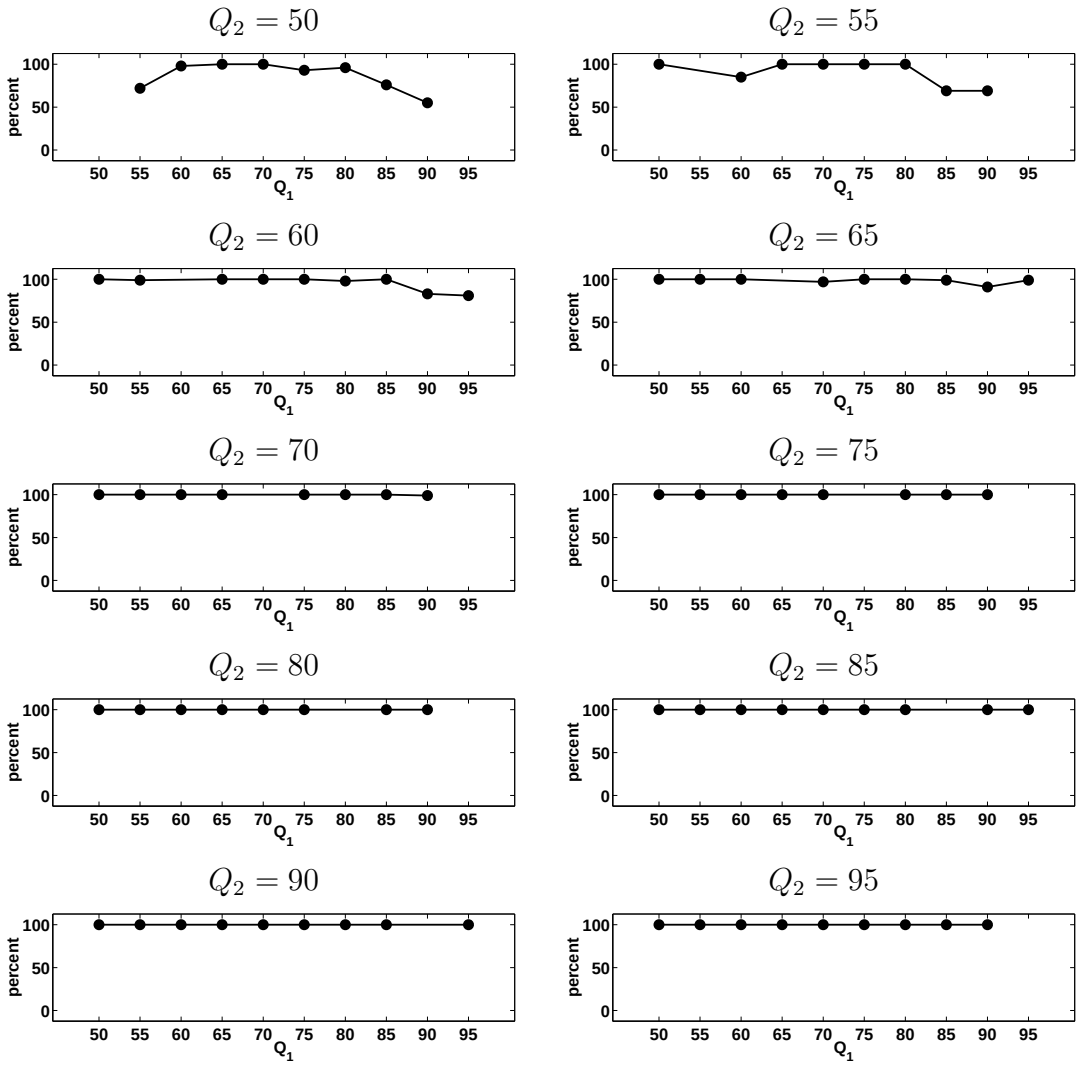

Figure 4.9: Shown are the accuracies of detecting double JPEG compressed images with a 0% false positive rate. Each graph shows the accuracies for a fixed second quality factor as a function of the first quality factor. The missing points in the graphs correspond to undetectable pairs of qualities. Each point corresponds to an average over 100 images. See also Table 4.1.

the double quantization is undetectable. Note also that for high first qualities and low second qualities factors, e.g., $Q_1 = 90$ and $Q_2 = 50$, the detection accuracy drops to approximately 50%. The reason for this decrease in accuracy is twofold. First, the quantization steps of the DCT coefficients are larger for images compressed with lower qualities (e.g., 50 to 65), and, as a result, the number of quantized DCT coefficients as a function of coefficient magnitude decays very fast, i.e., the DCT coefficient histograms have a support that is shorter and concentrated around zero. Second, double quantization with a small first step followed by a large second step (as is the case of images double compressed with a higher first quality followed by a lower second quality) introduces periodic artifacts with a large period, proportional to the second quantization step. Detecting periodic artifacts with a large period in a short support signal is, not surprisingly, more difficult.

4.4 Summary

We have proposed a technique for detecting if a JPEG image has been double compressed, as might occur if the image has been tampered with. This technique exploits the fact that double JPEG compression amounts to double quantization of the block DCT coefficients, which introduces specific artifacts visible in the histograms of these coefficients. We have devised a quantitative measure for these artifacts, and employed this measure to discriminate between single and double JPEG compressed images. There are certain theoretical limitations that make double compressed images impossible to detect in certain cases. We have, however, analyzed the sensitivity of the double JPEG detection method on a large number of images and found that a broad range of quality factors is detectable. We have noticed that images that are compressed first with a high quality, then with a significantly lower quality are generally harder to detect.

As is often the case with image authentication schemes, our method is vulnerable to attack. If a manipulated image in JPEG format is cropped[9] before being re-saved, the artifacts described in this chapter will no longer be present. It is possible that different artifacts may be present, and this will be an interesting problem for further research.

[9]The cropping must be done such that the grid of 8×8 image blocks used by the original JPEG compression is different from that used by the second compression. For example, if the top row of an image is removed, the 8×8 block grid is shifted down by one pixel, or if the leftmost column is removed the grid shifts right by one pixel. If the number of cropped rows or columns is a multiple of 8, however, the grids of the original and double compressed images remain the same.

4.5 Double JPEG Detection Algorithm

1. Given an image, JPEG compressed with quality Q, let $C_k(x, y)$ denote the coefficients of DCT frequency k, and $a(k)$ denote the step obtained from the quantization table stored in the JPEG file. The DCT frequencies are considered in the zig-zag order given by:

$$\begin{bmatrix} 1 & 3 & 4 & 10 & 11 & 21 & 22 & 36 \\ 2 & 5 & 9 & 12 & 20 & 23 & 35 & 37 \\ 6 & 8 & 13 & 19 & 24 & 34 & 38 & 49 \\ 7 & 14 & 18 & 25 & 33 & 39 & 48 & 50 \\ 15 & 17 & 26 & 32 & 40 & 47 & 51 & 58 \\ 16 & 27 & 31 & 41 & 46 & 52 & 57 & 59 \\ 28 & 30 & 42 & 45 & 53 & 56 & 60 & 63 \\ 29 & 43 & 44 & 54 & 55 & 61 & 62 & 64 \end{bmatrix}. \tag{4.27}$$

2. Initialize the parameters:

   - N: the number of DCT frequencies considered
   - V_k: the vectors of bin centers employed when computing histograms

3. For all k between 1 and N, compute first the histograms of the DCT coefficients $h_k(v)$, where $v \in V_k$. Next compute the periodicity measures $M(h_k; a(k))$, Equation (4.26). If $M(h_k, a(k)$ is greater than an empirically determined threshold, $T(k, a(k))$, then the coefficients, $C_k(x, y)$, are labeled as double quantized, else they are considered single quantized.

4. If at least one set of DCT coefficients, $C_k(x, y)$, is labeled as double quantized, then the image is classified as double JPEG compressed.

Chapter 5

Detection of Duplicated Image Regions

A common manipulation when tampering with an image is to copy and paste portions of the image to conceal a person or object in the scene. If the splicing is imperceptible, little concern is typically given to the fact that identical (or virtually identical) regions are present in the image. In this chapter, we present a technique that can efficiently detect and localize duplicated regions in an image. This technique works by first applying a principal component analysis (PCA) on small fixed-size image blocks to yield a reduced dimension representation. This representation is robust to minor variations in the image due to additive noise or lossy compression. Duplicated regions are then detected by lexicographically sorting all the image blocks. A similar method for detecting duplicated regions based on lexicographic sorting of DCT block coefficients was proposed in [24]. While both methods employ a similar approach, the data-driven PCA basis may better capture discriminating features. We show the efficacy of this technique on credible forgeries, and quantify its robustness and sensitivity to additive noise, and lossy compression schemes.

5.1 Detecting Duplicated Regions

Given an image with N pixels our task is to determine if it contains duplicated regions of unknown location and shape. An exhaustive approach that examines every possible pair of regions has an exponential complexity in the number of image pixels, and is obviously computationally prohibitive. A more efficient algorithm might look for duplication of small fixed-sized blocks[1]. By stringing each such block into a vector and lexicographically sorting all image blocks, identical blocks correspond to adjacent pairs in the sorted list. The primary cost of this algorithm is given by the lexicographic sorting, yielding a complexity in $O(N \log N)$, since the number of image blocks is proportional to N, the number of image pixels. Note that this is a significant improvement over the brute-force exponential algorithm. The drawback of this approach, however, is that it is sensitive to small variations between duplicated regions due to, for example, additive noise or lossy compression. We describe next an algorithm that overcomes these limitations while retaining its efficiency.

[1] We assume that the size of the blocks is considerably smaller than the duplicated region to be detected.

Consider a gray-scale image with N pixels (we discuss below how this algorithm extends to color images). An image is tiled with overlapping blocks of b pixels ($\sqrt{b} \times \sqrt{b}$ pixels in size), each of which is assumed to be considerably smaller than the size of the duplicated regions to be detected. Let $\vec{x}_i$, $i = 1, \ldots, N_b$, denote these blocks in vectorized form, where $N_b = (\sqrt{N} - \sqrt{b} + 1)^2$. We now consider an alternate representation of these image blocks based on a principal component analysis (PCA) [15]. Assume that the blocks $\vec{x}_i$ are zero-mean[2], and compute the covariance matrix as:

$$C = \sum_{i=1}^{N_b} \vec{x}_i \vec{x}_i^T. \tag{5.1}$$

The orthonormal eigenvectors, $\vec{e}_j$, of the matrix C, with corresponding eigenvalues, λ_j, satisfying:

$$C\vec{e}_j = \lambda_j \vec{e}_j, \tag{5.2}$$

define the principal components, where $j = 1, \ldots, b$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_b$. The eigenvectors, $\vec{e}_j$, form a new linear basis for each image block, $\vec{x}_i$:

$$\vec{x}_i = \sum_{j=1}^{b} a_j \vec{e}_j, \tag{5.3}$$

where $a_j = \vec{x}_i^T \vec{e}_j$, and $\vec{a}_i = (a_1 \ \ldots \ a_b)$ is the new representation for each image block.

The dimensionality of this representation can be reduced by simply truncating the sum in Equation (5.3) to the first N_t terms. Note that the projection onto the first N_t eigenvectors of the PCA basis gives the best N_t-dimensional approximation in the least squares sense, if the distribution of the vectors, $\vec{x}_i$, is a multi-dimensional Gaussian [15]. This reduced dimension representation, therefore, provides a convenient space in which to identify similar blocks in the presence of corrupting noise, as truncation of the basis will remove minor intensity variations.

The detection algorithm proceeds as follows. First, to further reduce minor variations due to corrupting noise, the reduced dimension representation of each image block, $\vec{a}_i$, is component-wise quantized, $\lfloor \vec{a}_i / Q \rfloor$, where the positive integer Q denotes the number of quantization bins[3]. A $N_b \times b$ matrix is constructed whose rows contain these quantized coefficients. Let the matrix S be the result of lexicographically sorting the rows of this matrix in column order. Let $\vec{s}_i$ denote the i^{th} row of this sorted matrix, and let the tuple (x_i, y_i) denote the block's image coordinates (top-left corner) that corresponds to $\vec{s}_i$. Consider next all pairs of rows $\vec{s}_i$ and $\vec{s}_j$, whose row distance, $|i-j|$, in the sorted matrix S is less than a specified threshold. The offset, in the image, of all such pairs is given by:

$$\begin{aligned} (x_i - x_j, y_i - y_j) &\quad \text{if} \quad x_i - x_j > 0 \\ (x_j - x_i, y_i - y_j) &\quad \text{if} \quad x_i - x_j < 0 \\ (0, |y_i - y_j|) &\quad \text{if} \quad x_i = x_j \end{aligned}$$

[2]If the blocks, $\vec{x}_i$, are not zero-mean, then the mean, $\vec{\mu} = 1/N_b \sum_{i=1}^{N_b} \vec{x}_i$, should be subtracted from each block, $\vec{x}_i - \vec{\mu}$.

[3]For simplicity we a use a constant number of quantization bins, although it might be more appropriate to use more bins for the coordinates with higher variance, and fewer bins for the lower variance coordinates.

From a list of all such offsets, duplicated regions in the image are detected by noting the offsets with high occurrence. For example, a large duplicated region will consist of many smaller blocks, each of these blocks will appear in close proximity to each other in the lexicographically sorted matrix, and will have the same offset. In order to avoid false hits due to uniform intensity areas, offset magnitudes below a specified threshold are ignored. See Section 5.4 for a detailed step-by-step algorithm.

The results of this detection can be visualized by constructing a duplication map — a zero image of the same size as the original is created, and all pixels in a region believed to be duplicated are assigned a unique gray-scale value. The complexity of this algorithm, dominated by the lexicographic sorting, is in $O(N_t N \log N)$, where N_t is the dimension of the PCA reduced representations and N is the total number of image pixels.

There are at least two ways in which this algorithm can be extended to color images. The simplest approach is to independently process each color channel (e.g., RGB) to yield three duplication maps. The second approach is to apply PCA to color blocks of size $3b$, and proceed in the same way as described above. The duplication detection algorithm can also be employed to detect if two, or more, images contain identical regions, as would happen when a region from one image is copied then pasted into a different image. Given two images, the PCA is applied to overlapping blocks from both images, and the detection proceeds as in the single image case.

5.2 Results

Shown in Figures 5.1-5.3 are original and tampered images. The tampering consisted of copying and pasting a region in the image to conceal a person or object. Shown in the lower portion of these figures are the outputs of our detection algorithm as applied to the tampered image saved with JPEG quality factors between 50 and 100. In each map, the two duplicated regions are shown with different gray-scale values. In all of these examples, all parameters were fixed and set to: $b = 64$, $\epsilon = 0.01$, $Q = 256$, $N_n = 100$, $N_f = 128$, $N_d = 16$ (see Section 5.4 for details). Truncation of the PCA basis typically reduces the dimension from 64 to 32. The average runtime for one color channel of a 512×512 image running on a 3 GHz processor, is approximately 10 seconds. For visualization purposes, the duplication map was (1) dilated then eroded to eliminate holes in the duplicated regions, and (2) eroded then dilated to eliminate spurious pairs of duplicated blocks. A disk-shaped structuring element with a radius of 20 pixels was employed for these morphologic operations [26].

To quantify the robustness and sensitivity of our algorithm we constructed a database of 100 color images of size 512×512 pixels. These images were cropped from larger 2000×3008 images taken with a Nikon D100 digital camera. In each image, a random square region was copied and pasted onto a random, non-overlapping position in the image. Each image was then either JPEG or JPEG2000 compressed with varying quality factors, or corrupted with additive noise with varying signal to noise ratios (SNR). Shown on the top row of Figure 5.4, for example, are four images with duplicated regions of size 32×32, 64×64, 96×96, and 128×128 — the first two images were compressed with JPEG qualities 85 and 65, and the other two images were corrupted with additive

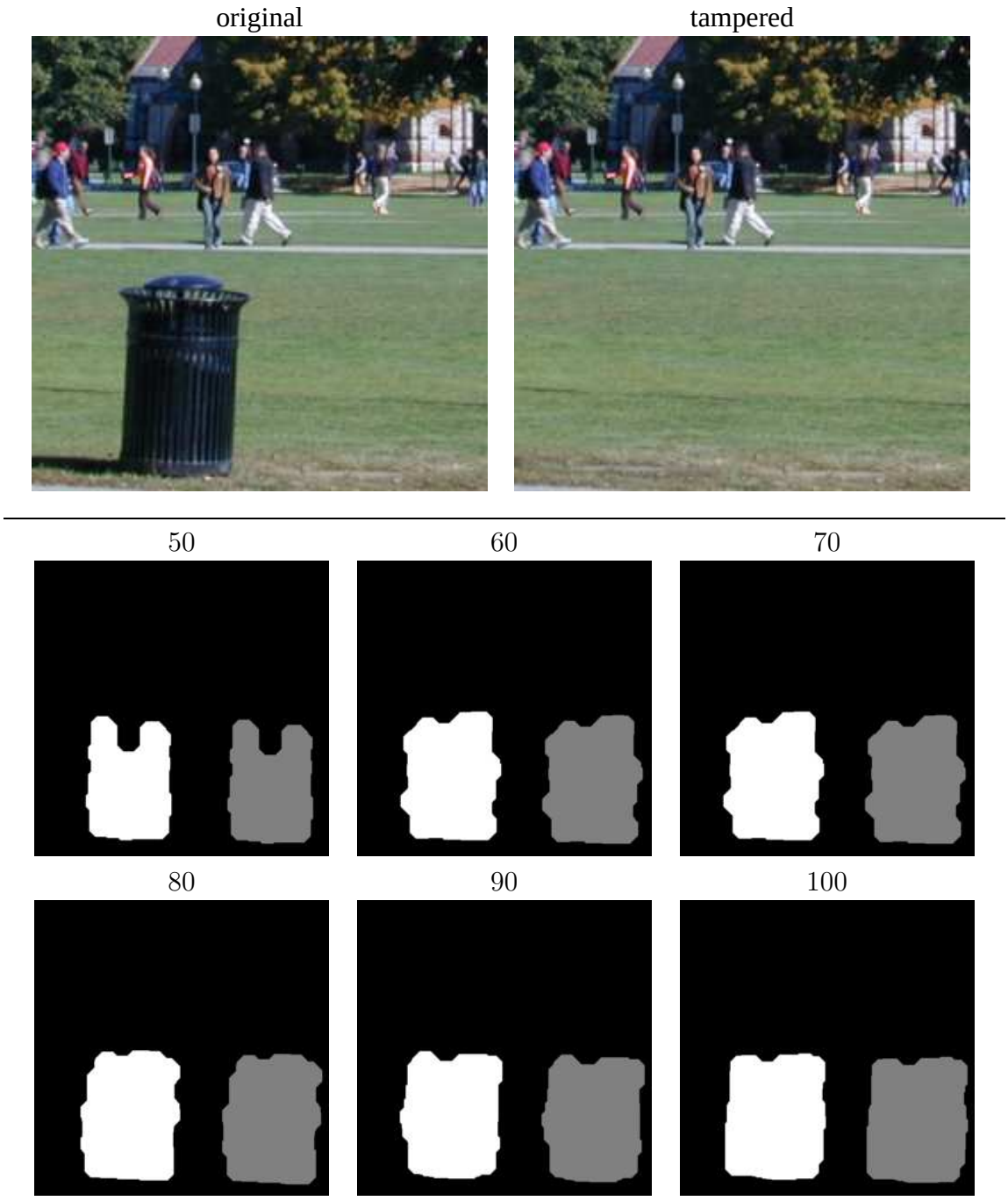

Figure 5.1: Shown are an original and a tampered image. Shown below are the output duplication maps from the green channel of the tampered image saved with JPEG qualities ranging between 50 and 100.

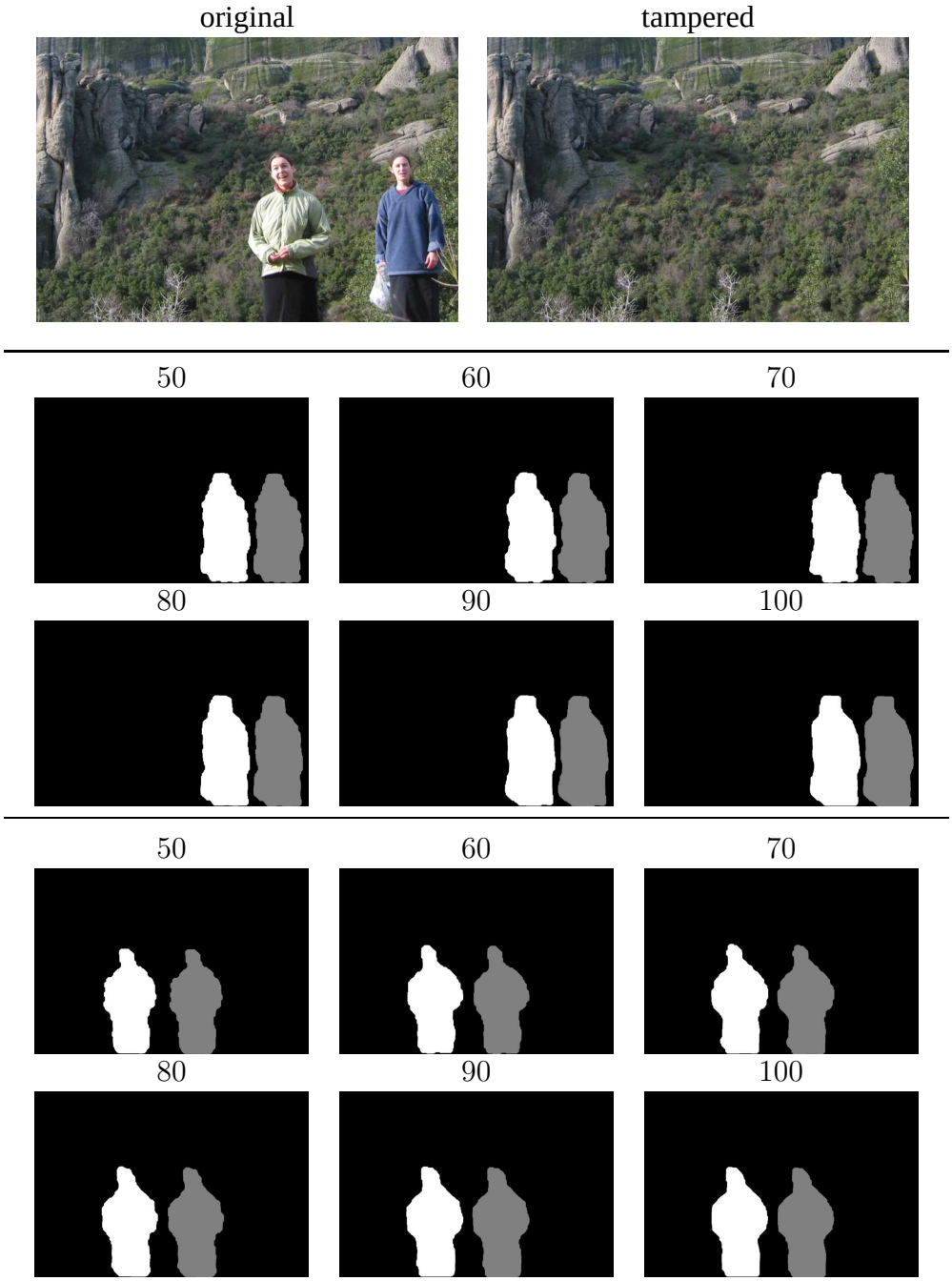

Figure 5.2: Shown are an original and a tampered image. Shown below are the output duplication maps (corresponding to different regions used to conceal each person) from the green channel of the tampered image saved with JPEG qualities ranging between 50 and 100.

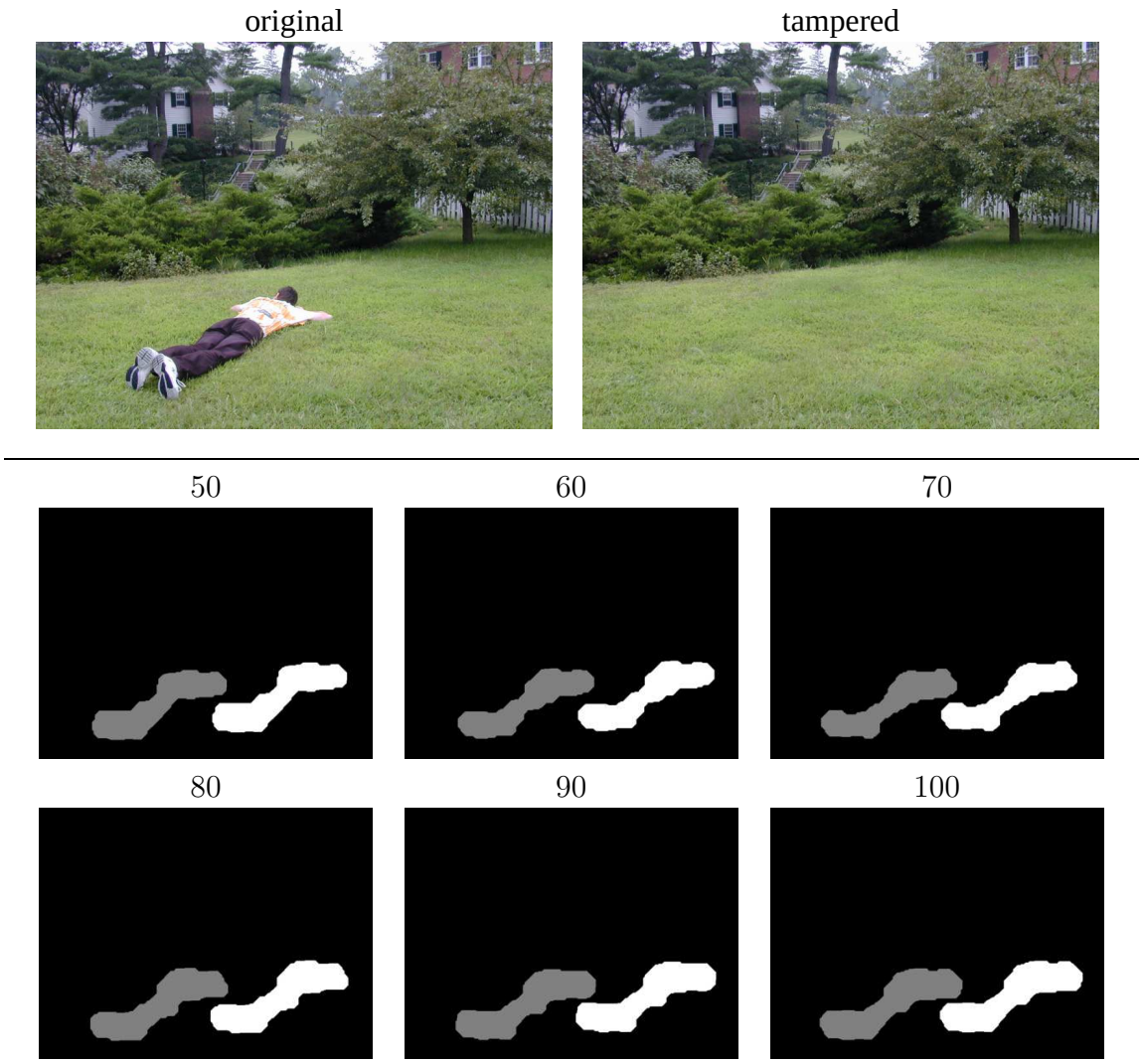

Figure 5.3: Shown are an original and a tampered image. Shown below are the output duplication maps from the green channel of the tampered image saved with JPEG qualities ranging between 50 and 100.

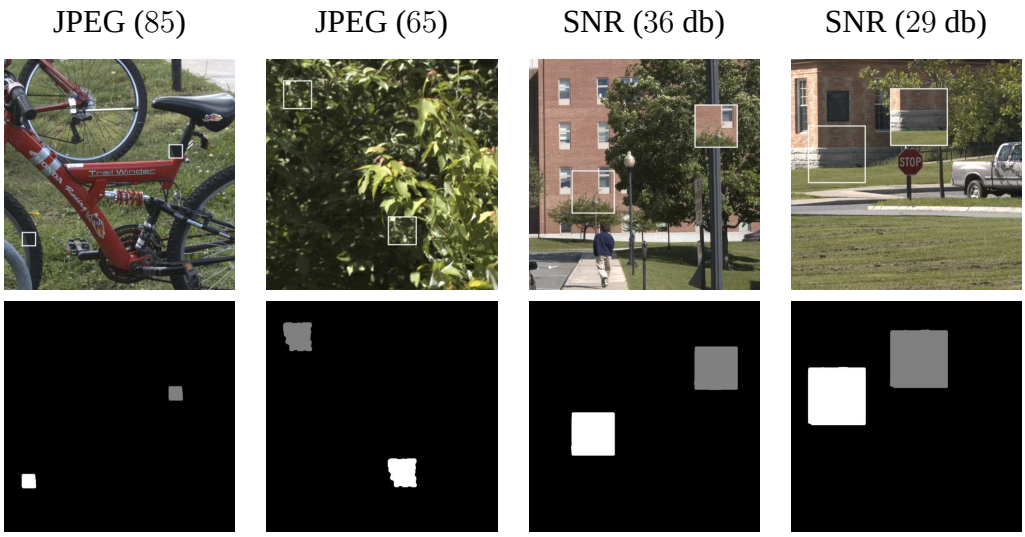

Figure 5.4: Shown on the top row are four images with duplicated regions of size 32×32, 64×64, 96×96, and 128×128, after having been JPEG compressed (first and second images) or corrupted with additive noise (third and fourth images). Shown below are the duplication maps returned when running our algorithm on each image's green channel.

noise with SNRs of 36db and 29db. Shown on the bottom row of Figure 5.4 are the duplication maps returned when running our algorithm on the green channel of each image. In these examples, and those described below, all parameters were set to: $b = 64$, $\epsilon = 0.01$, $Q = 256$, $N_n = 100$, $N_f = 128$, $N_d = 16$. Shown in Figure 5.5 are the detection accuracy and the false positive rate as functions of JPEG compression quality for different sizes of duplicated regions. Note that the accuracy is, in general, very good, except for small block sizes and low JPEG qualities. Note also that the average number of false positives (regions incorrectly labeled as duplicated) is relatively low. Shown in Figure 5.6 are the detection accuracy and the false positive rate as functions of the JPEG2000 compression quality (in bits per pixel). The accuracy is nearly perfect for block sizes greater than 32×32, and the false positive rate is low. When compared to classic JPEG, the only notable difference is that, in JPEG2000 compressed images, the detection of 64×64 duplicated regions is more robust (higher accuracy for lower qualities), while the detection of 32×32 duplicated regions is less robust (lower accuracy for higher qualities). Shown in Figure 5.7 are the detection accuracy and false positive rate as functions of signal to noise ratio (SNR) of additive white Gaussian noise. As in the previous examples, the detection rates are nearly perfect, except for small block sizes and low SNR. Non-linear point-wise transformations, e.g., gamma correction, do not affect the detection of duplicated regions in an image as long as they are applied uniformly across the image. The same is true for GIF compression. Since identical color pixels in the original image map to the same indexed color in the GIF image, duplicated regions in the original image remain identical in the GIF image.

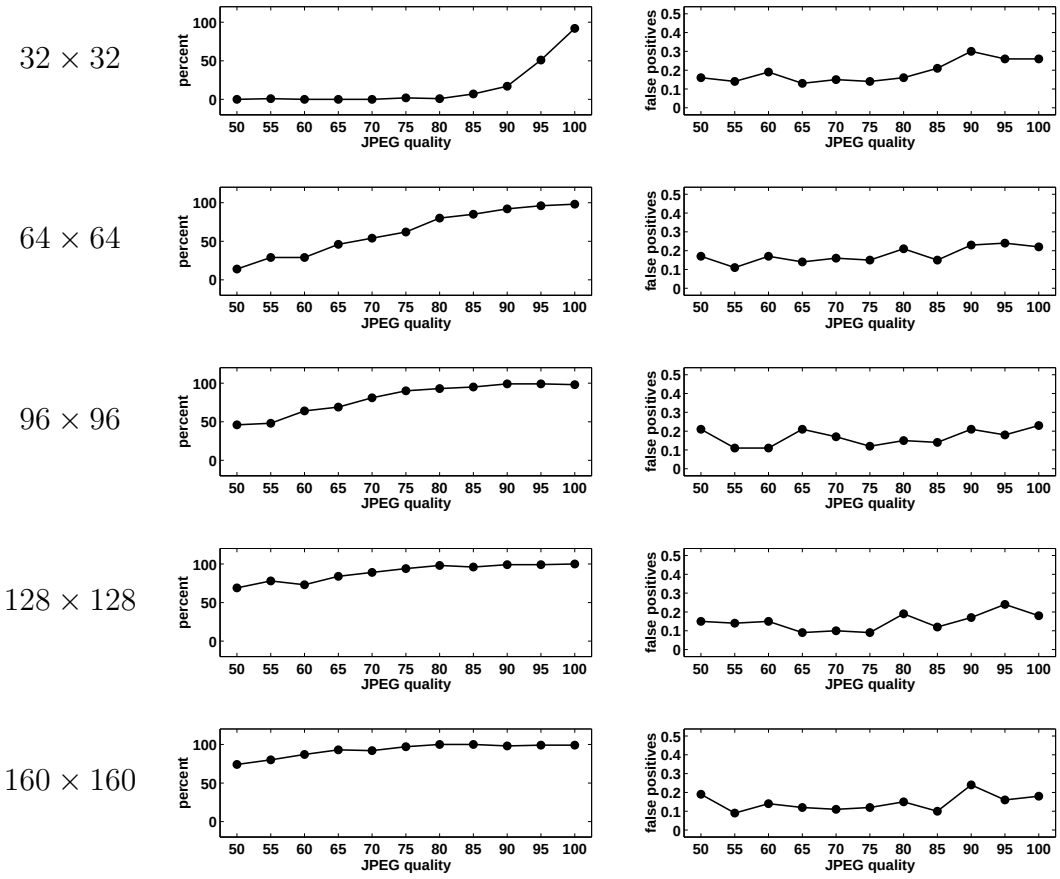

Figure 5.5: The detection accuracies (left column) and the average number of false positives (right column) are shown as functions of the JPEG compression quality. Each row corresponds to duplicated blocks of size ranging from 32×32 to 160×160 pixels. Each data point corresponds to an average over 100 images.

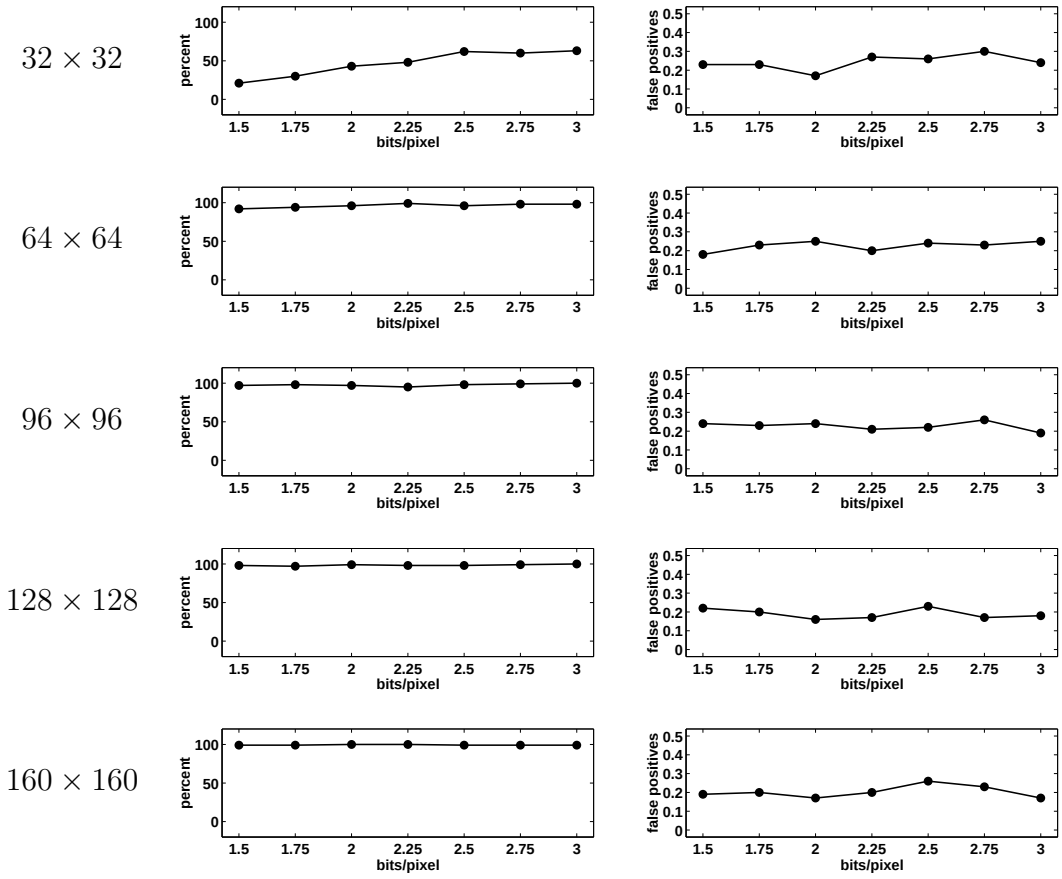

Figure 5.6: The detection accuracies (left column) and the average number of false positives (right column) are shown as functions of the JPEG2000 compression quality (in bits per pixel). Each row corresponds to duplicated blocks of size ranging from 32×32 to 160×160 pixels. Each data point corresponds to an average over 100 images.

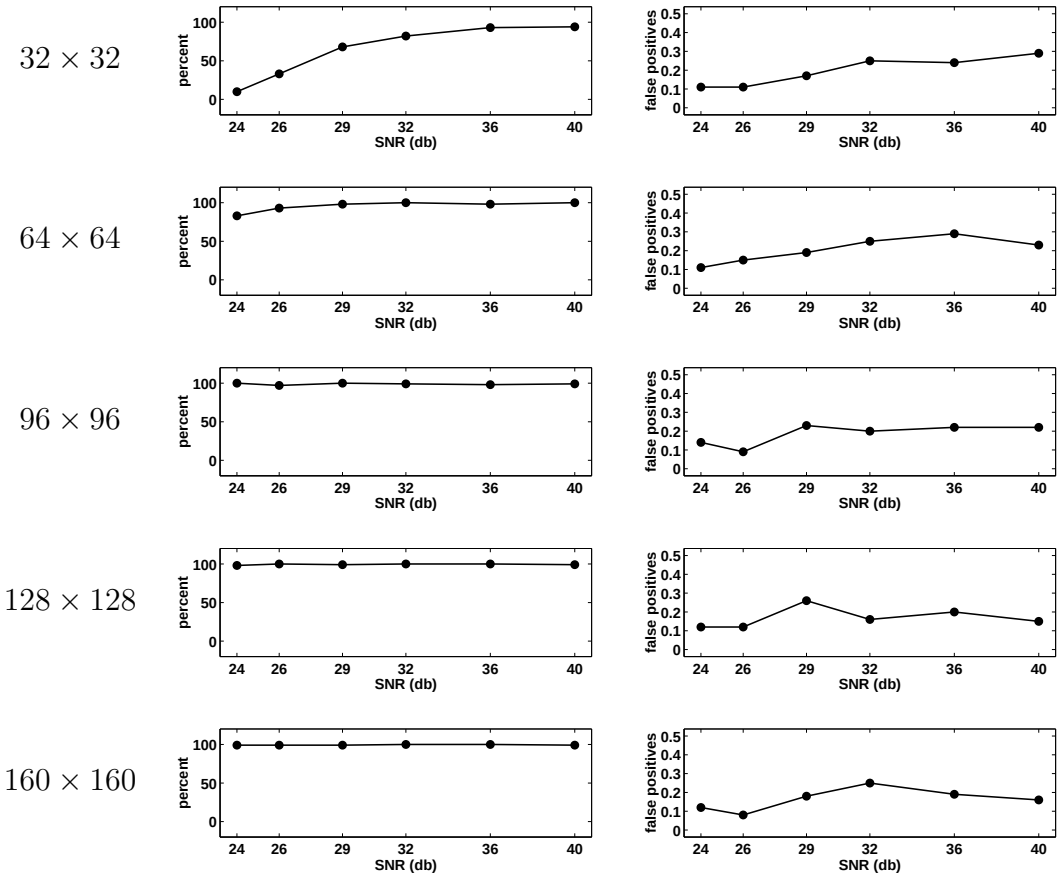

Figure 5.7: The detection accuracies (left column) and the average number of false positives (right column) are shown as functions of the SNR of added white Gaussian noise. Each row corresponds to duplicated blocks of sizes ranging from 32×32 to 160×160 pixels. Each data point corresponds to an average over 100 images.

5.3 Summary

We have presented an efficient and robust technique that automatically detects duplicated regions in an image. This technique works by first applying a principal component analysis (PCA) on small, fixed-size image blocks to yield a reduced dimension representation that is robust to minor variations in the image due to additive noise or lossy compression. Duplicated regions are then detected by lexicographically sorting all the image blocks. We have shown the effectiveness of this technique on plausible forgeries, and have quantified its sensitivity to additive noise, and lossy compression schemes — we find that detection is possible even in the presence of significant amounts of distortion.

5.4 Duplication Detection Algorithm

1. Let N be the total number of pixels in a gray-scale or color image
2. Initialize the parameters:
   - b: number of pixels per block ($\sqrt{b} \times \sqrt{b}$ pixels in dimension) — there are $N_b = (\sqrt{N} - \sqrt{b} + 1)^2$ such blocks
   - ϵ: fraction of the ignored variance along the principal axes
   - Q: number of quantization bins
   - N_n: number of neighboring rows to search in lexicographically sorted matrix
   - N_f: minimum frequency threshold
   - N_d: minimum offset threshold
3. Using PCA, compute the new N_t-dimensional representation, $\vec{a}_i$, $i = 1, \ldots, N_b$, of each b pixel image block (for color images: (1) analyze each color channel separately; or (2) build a single color block of size $3b$ pixels). The value of N_t is chosen to satisfy: $1 - \epsilon = \frac{\sum_{i=1}^{N_t} \lambda_i}{\sum_{i=1}^{b} \lambda_i}$, where λ_i are the eigenvalues as computed by the PCA, Equation (5.2).
4. Build a $N_b \times b$ matrix whose rows are given by the component-wise quantized coordinates: $\lfloor \vec{a}_i / Q \rfloor$.
5. Sort the rows of the above matrix in lexicographic order to yield a matrix S. Let $\vec{s}_i$ denote the rows of S, and let (x_i, y_i) denote the coordinates (top-left corner in the image) of the block that corresponds to $\vec{s}_i$.
6. For every pair of rows $\vec{s}_i$ and $\vec{s}_j$ from S such that $|i - j| < N_n$, place the pair of coordinates (x_i, y_i) and (x_j, y_j) onto a list.
7. For all elements in this list, compute their offsets, defined as:

$$\begin{aligned} (x_i - x_j, y_i - y_j) \quad & \text{if} \quad x_i - x_j > 0 \\ (x_j - x_i, y_i - y_j) \quad & \text{if} \quad x_i - x_j < 0 \\ (0, |y_i - y_j|) \quad & \text{if} \quad x_i = x_j \end{aligned}$$

8. Discard all pairs of coordinates with an offset frequency less than N_f.
9. Discard all pairs whose offset magnitude, $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, is less than N_d.
10. From the remaining pairs of blocks build a duplication map by constructing a zero image of the same size as the original, and coloring all pixels in a duplicated region with a unique gray-scale intensity value.

Chapter 6

Blind Estimation of Background Noise

Digital images have an inherent amount of noise introduced either by the imaging process or by digital compression. The amount of noise is typically uniform across the entire image. If two images with different noise levels are spliced together, or if small amounts of noise are locally added to conceal traces of tampering, then variations in the local noise variance across the image can be used as evidence of tampering. Measuring the local noise variance is non-trivial in the absence of the original signal. Several *blind* signal to noise ratio (SNR) estimators have, however, been proposed [49]. Since these estimators work by estimating the variances of both the signal and the noise, they can be employed to measure the local noise variance of an image[1]. We present one such estimator, M_2M_4 [43], and show its effectiveness in measuring the local noise variance in images.

6.1 Blind Signal-to-Noise Ratio Estimation

We begin by assuming an additive noise model:

$$y[t] = x[t] + w[t], \tag{6.1}$$

where $x[t]$ is the uncorrupted signal with variance S and $w[t]$ is the noise with variance N. Let $M_2 = \mathcal{E}\{y^2[t]\}$ and $M_4 = \mathcal{E}\{y^4[t]\}$ denote the second and forth moments of the corrupted signal, where $\mathcal{E}\{\cdot\}$ is the expected value operator[2]. Assuming that the signal and noise are independent

[1] To be invariant to the underlying signal strength, we analyze the noise variance instead of the ratio of signal to noise variances.

[2] In practice, all the expected values are estimated as sample averages. For example, the second and forth moments of the noisy signal are computed as: $M_2 = \frac{1}{N}\sum_{t=1}^{N}(y[t]-\mu)^2$, and $M_4 = \frac{1}{N}\sum_{t=1}^{N}(y[t]-\mu)^4$, where $\mu = \frac{1}{N}\sum_{t=1}^{N} y[t]$ denotes the sample mean.

and zero-mean, it was shown [49] that M_2 and M_4 take the form:

$$\begin{aligned} M_2 &= \mathcal{E}\left\{y^2[t]\right\} \\ &= \mathcal{E}\left\{(x[t]+w[t])^2\right\} \\ &= \mathcal{E}\left\{x^2[t]\right\} + 2\,\mathcal{E}\left\{x[t]w[t]\right\} + \mathcal{E}\left\{w^2[t]\right\} \\ &= S + N \end{aligned} \tag{6.2}$$

$$\begin{aligned} M_4 &= \mathcal{E}\left\{y^4[t]\right\} \\ &= \mathcal{E}\left\{(x[t]+w[t])^4\right\} \\ &= \mathcal{E}\left\{x^4[t]\right\} + 4\,\mathcal{E}\left\{x^3[t]w[t]\right\} + 6\,\mathcal{E}\left\{x^2[t]w^2[t]\right\} \\ &\quad + 4\,\mathcal{E}\left\{x[t]w^3[t]\right\} + \mathcal{E}\left\{w^4[t]\right\} \\ &= k_x S^2 + 6SN + k_w N^2, \end{aligned} \tag{6.3}$$

where $k_x = \mathcal{E}\left\{x^4[t]\right\}/(\mathcal{E}\left\{x^2[t]\right\})^2$ and $k_w = \mathcal{E}\left\{w^4[t]\right\}/(\mathcal{E}\left\{w^2[t]\right\})^2$ are the kurtosis's values of the original signal and noise. Solving Equations (6.2-6.3) for S and N yields the estimators:

$$\hat{S} = \frac{M_2(k_w-3) \pm \sqrt{(9-k_x k_w)M_2^2 + M_4(k_x+k_w-6)}}{k_x+k_y-6} \tag{6.4}$$

$$\hat{N} = M_2 - \hat{S}. \tag{6.5}$$

Assuming white Gaussian noise ($k_w = 3$), and denoting the kurtosis of the noisy signal with $k_y = M_4/M_2^2$, the Equations (6.4-6.5) become:

$$\hat{S} = M_2\sqrt{\frac{k_y-3}{k_x-3}} \tag{6.6}$$

$$\hat{N} = M_2\left(1-\sqrt{\frac{k_y-3}{k_x-3}}\right). \tag{6.7}$$

Note that these estimators work only when the original signal is *not* Gaussian, i.e., $k_x \neq 3$. Note also that in order to obtain reasonable estimates from Equations (6.6-6.7), i.e., positive real values, the following condition must hold:

$$0 \leq \frac{k_y-3}{k_x-3} \leq 1. \tag{6.8}$$

This condition is equivalent to saying that the kurtosis of the noisy signal, k_y must be between the kurtosis of the original signal, k_x, and three, the kurtosis of the noise. Due to numerical errors and to the fact that instead of the true kurtosis values, sample average estimators are employed, the above conditions may not be always fulfilled, and complex or negative values may occur on occasion.

Note that for these estimators the kurtosis of the original signal is assumed to be known, which may not be true in general. In the results presented below, it is assumed to be known. It may be possible, for example, to estimate the original kurtosis from a region of an image that is believed to be unadulterated.

Figure 6.1: Two images taken with a Nikon D100 camera. See also Figures 6.2-6.3.

6.2 Results

We have employed the blind estimator presented in the previous section, Equation (6.7), to measure the local noise variance in an image. We start by dividing the image into overlapping blocks, then, for each block, the noise variance is estimated. If the estimates reveal two or more different levels of noise, it is highly probable that the image has been tampered with.

Shown in Figure 6.1 are two color images of resolutions 3008×2000 (left) and 1200×798 (right), taken with a Nikon D100 digital camera. White Gaussian noise with different variances was locally added to these images, and the blind estimator has been applied on 64×64 overlapping[3] blocks of the green channel. Shown in Figures 6.2-6.3 in the left and middle columns are the green channels of the noisy images and the absolute differences between the original and the noisy channels. The absolute differences are shown on the same intensity scale. Note that the intensity of the noisy region in the absolute difference images is proportional to the amount of noise, i.e., it decreases when the noise variance decreases. In the right column, the estimated local noise variances are shown as images on the same logarithmic intensity scale (the intensity of each pixel is proportional to the logarithm of the estimated noise variance in a 64×64 block). Note that the estimated noise variance's logarithm is visibly higher in the noisy region, and it decreases

[3]The overlap was 32 pixels along each direction.

32×32

actual	estimated (db)			
(db)	mean	std	min	max
-6.00	-6.045	0.255	-6.535	-5.411
-5.50	-5.608	0.227	-6.018	-5.018
-5.00	-5.138	0.179	-5.428	-4.675
-4.50	-4.617	0.127	-4.805	-4.259
-4.00	-4.059	0.079	-4.205	-3.846
-3.50	-3.490	0.049	-3.612	-3.390
-3.00	-2.947	0.040	-3.077	-2.862
-2.50	-2.465	0.051	-2.606	-2.331
-2.00	-2.083	0.076	-2.263	-1.884

64×64

actual	estimated (db)			
(db)	mean	std	min	max
-6.00	-5.993	0.269	-6.437	-5.276
-5.50	-5.568	0.226	-5.910	-4.973
-5.00	-5.095	0.166	-5.344	-4.644
-4.50	-4.578	0.118	-4.747	-4.207
-4.00	-4.020	0.077	-4.119	-3.752
-3.50	-3.469	0.057	-3.547	-3.260
-3.00	-2.938	0.048	-3.009	-2.775
-2.50	-2.457	0.055	-2.542	-2.271
-2.00	-2.063	0.090	-2.266	-1.813

96×96

actual	estimated (db)			
(db)	mean	std	min	max
-6.00	-5.990	0.286	-6.432	-5.323
-5.50	-5.566	0.242	-5.962	-5.000
-5.00	-5.092	0.177	-5.325	-4.625
-4.50	-4.567	0.122	-4.716	-4.182
-4.00	-4.018	0.086	-4.117	-3.724
-3.50	-3.469	0.065	-3.536	-3.221
-3.00	-2.948	0.056	-3.062	-2.734
-2.50	-2.466	0.064	-2.668	-2.244
-2.00	-2.055	0.096	-2.268	-1.797

128×128

actual	estimated (db)			
(db)	mean	std	min	max
-6.00	-5.922	0.307	-6.395	-5.144
-5.50	-5.514	0.250	-5.871	-4.819
-5.00	-5.045	0.186	-5.285	-4.460
-4.50	-4.527	0.138	-4.705	-4.080
-4.00	-3.993	0.093	-4.120	-3.657
-3.50	-3.458	0.068	-3.534	-3.193
-3.00	-2.948	0.059	-3.081	-2.738
-2.50	-2.467	0.065	-2.674	-2.295
-2.00	-2.049	0.100	-2.299	-1.795

Table 6.1: Shown are the statistics of the estimated noise variances (mean, standard deviation, and minimum and maximum values) as obtained for different block sizes. These statistics have been obtained from 50 natural images. On average, the correct value is estimated with 1.5% of the actual value. See also Figure 6.4.

proportionally with the amount of noise.

6.2.1 Sensitivity and Robustness

To quantify the sensitivity of the blind noise variance estimation, we have employed a database of 100 color images 1024×1024 in size. These images were cropped from a smaller set of 30 images 3008×2000 pixels in size, taken with a Nikon D100 camera.

Single-Channel Gray-Scale Images

We have employed the green channels of the images in our database as single-channel, gray-scale images. White Gaussian noise with different variances was uniformly added to these gray-scale images. Each image was divided into overlapping blocks (with an overlap of 50% of the block size

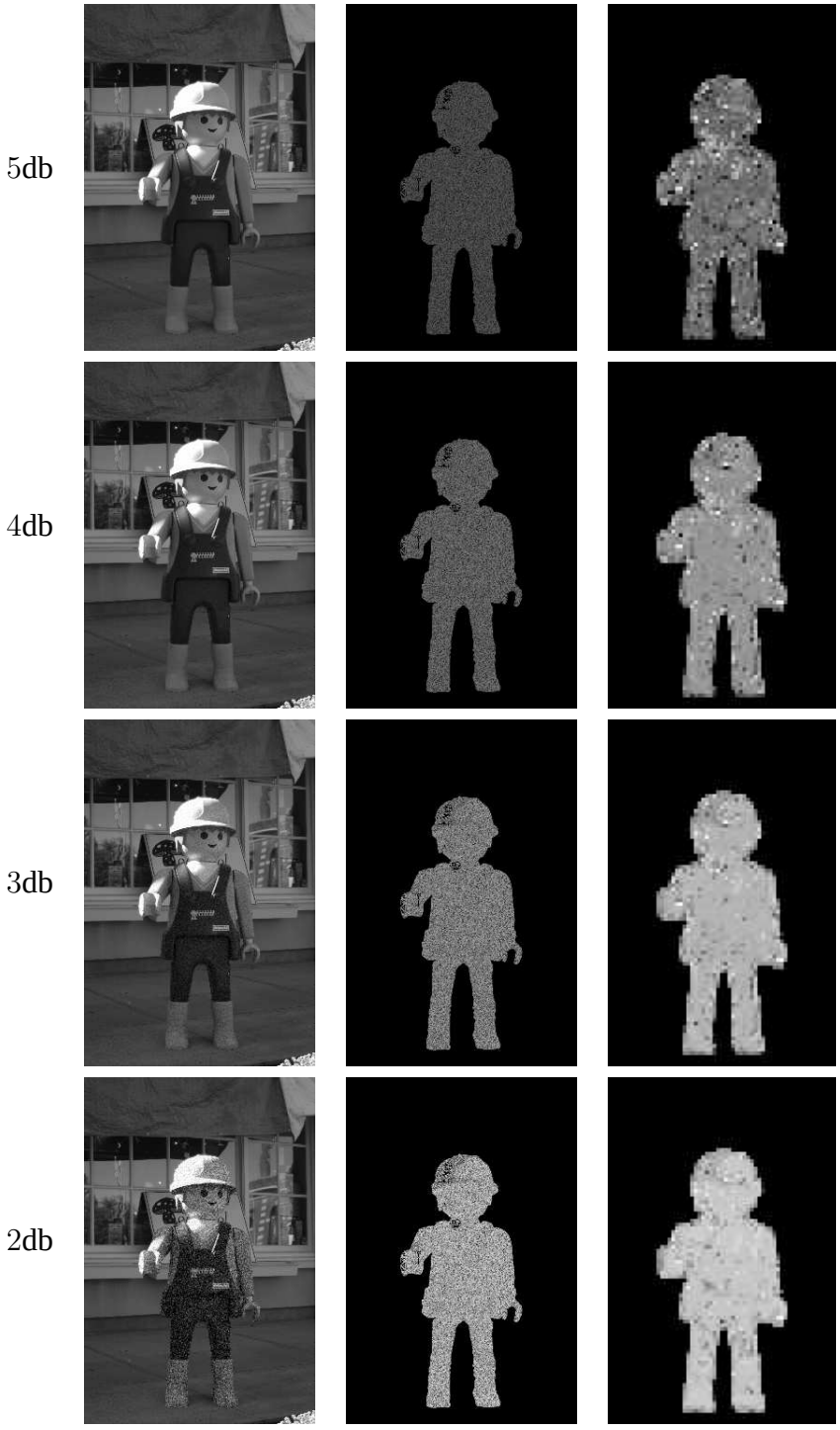

Figure 6.2: Shown on the left are the green channels of the left image in Figure 6.1 to which white Gaussian noise with different variances was locally added, in the middle are the absolute differences between the noisy and noiseless green channels, and on the right are the results of the local noise variance estimation.

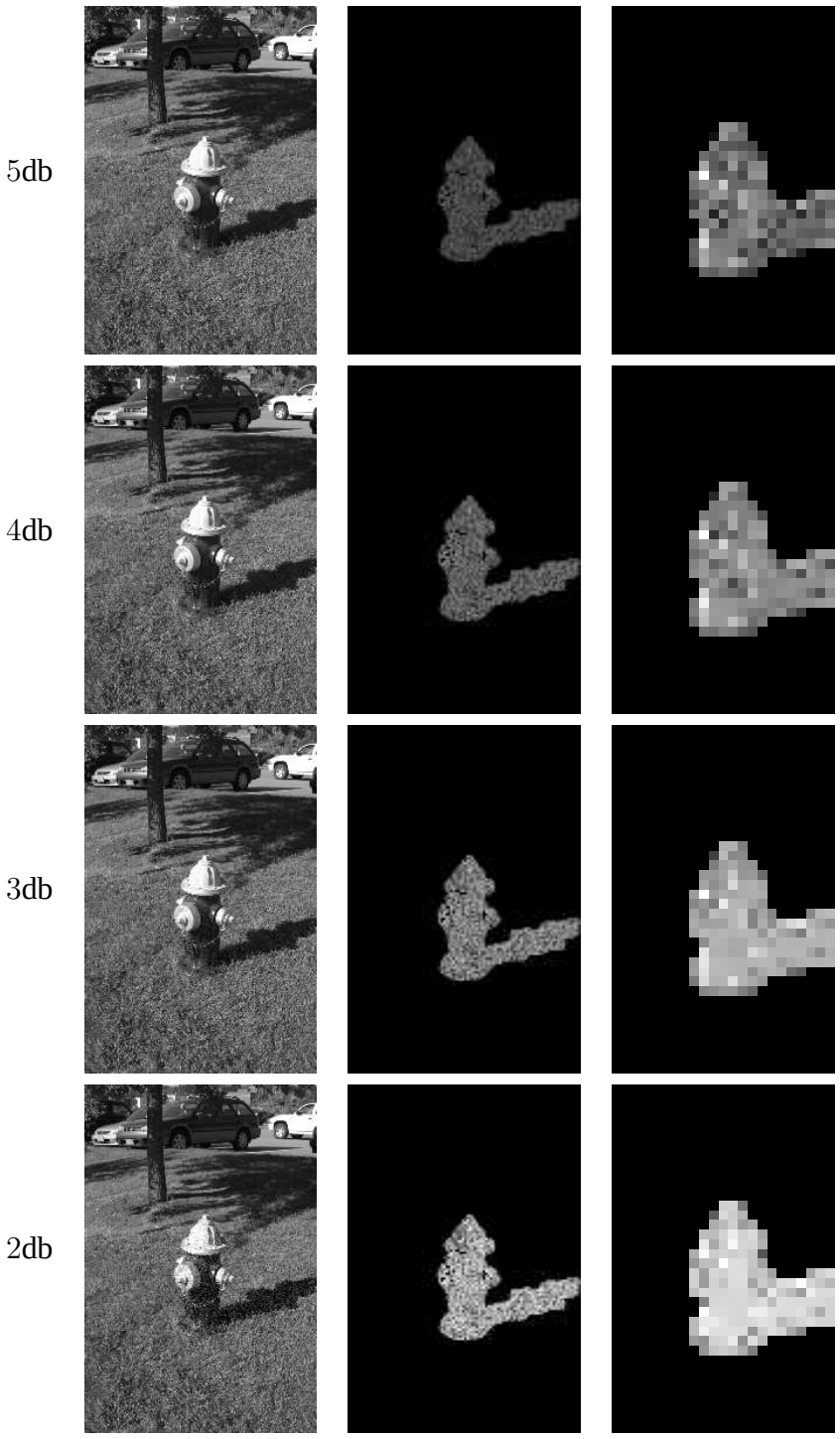

Figure 6.3: Shown on the left are the green channels of the right image in Figure 6.1 to which white Gaussian noise with different variances was locally added, in the middle are the absolute differences between the noisy and noiseless green channels, and on the right are the results of the local noise variance estimation.

along each direction), and noise variance estimates were computed using the estimator[4] in Equation (6.7). A global estimate was computed by first discarding the complex or negative individual estimates, then averaging the remaining ones. The results of blindly estimating the noise variance from 50 gray-scale images are shown in Table 6.1, and summarized in Figure 6.4. The original and estimated values of the noise variance are shown in decibels[5] (db). On average, the correct value is estimated within 1.5% of the actual value. Note that the estimator is more precise (i.e., smaller standard deviation) for larger noise variances, e.g., more than -4db. This is due to numerical and estimation errors which become more important for small noise variances.

We have noticed that due to numerical errors and errors in the estimation of the kurtosis, a constant bias is introduced in the estimated noise variances. This bias consists in an over-estimation of noise variances when the true values are very small, i.e., less than -4db. We have also noticed that the bias is dependent of the block size used in the estimation. As such, in all our quantitative results, the estimated noise variance, $\hat{N}$, is related to the true value, N, by the following empirically determined, quadratic relationships:

block size	quadratic relationship
32×32	$N = -0.124890\,\hat{N}^2 + 0.36134\,\hat{N} - 0.67819$
64×64	$N = -0.083903\,\hat{N}^2 + 0.55407\,\hat{N} - 0.51599$
96×96	$N = -0.066877\,\hat{N}^2 + 0.64985\,\hat{N} - 0.40843$
128×128	$N = -0.052028\,\hat{N}^2 + 0.72881\,\hat{N} - 0.31318$

These relationships were obtained using estimates from 50 gray-scale images different from the ones employed in our sensitivity experiments. Note that, when the block size increases, the coefficients of $\hat{N}$ increase towards one, while the coefficients of $\hat{N}^2$ and the free terms decrease toward zero. This suggests that the bias is becoming less pronounced for bigger window sizes. There is, however, a trade-off between the block size and the resolution of the noise variance estimates, which makes very large block sizes impractical. In our experiments, we have found that a 64×64 block size offered the best trade-off.

Multi-Channel Color Images

The main drawback, when blindly estimating the local noise variance in a single-channel image, is that the local kurtosis values of the noiseless image need to be known. In the case of multi-channel, color images it is possible to employ a simple heuristic to estimate the original kurtosis values. Under the assumption that the noise has the same distribution across all channels, this heuristic works by first taking the average of the color channels (red, green, blue), then estimating the local kurtosis from this average channel to approximate the original local kurtosis of the image's green channel. The estimation then proceeds as before: the green channel is divided into overlapping blocks, and the noise variance of each block is computed using the estimator in Equation 6.7, and the previously computed estimates of the original local kurtosis.

[4] In all experiments we have employed the sample average estimates from the original green channels of the images as original kurtosis values.

[5] If N denotes the variance of a noise signal, then its value in decibels is given by: $10 \log_{10} N$.

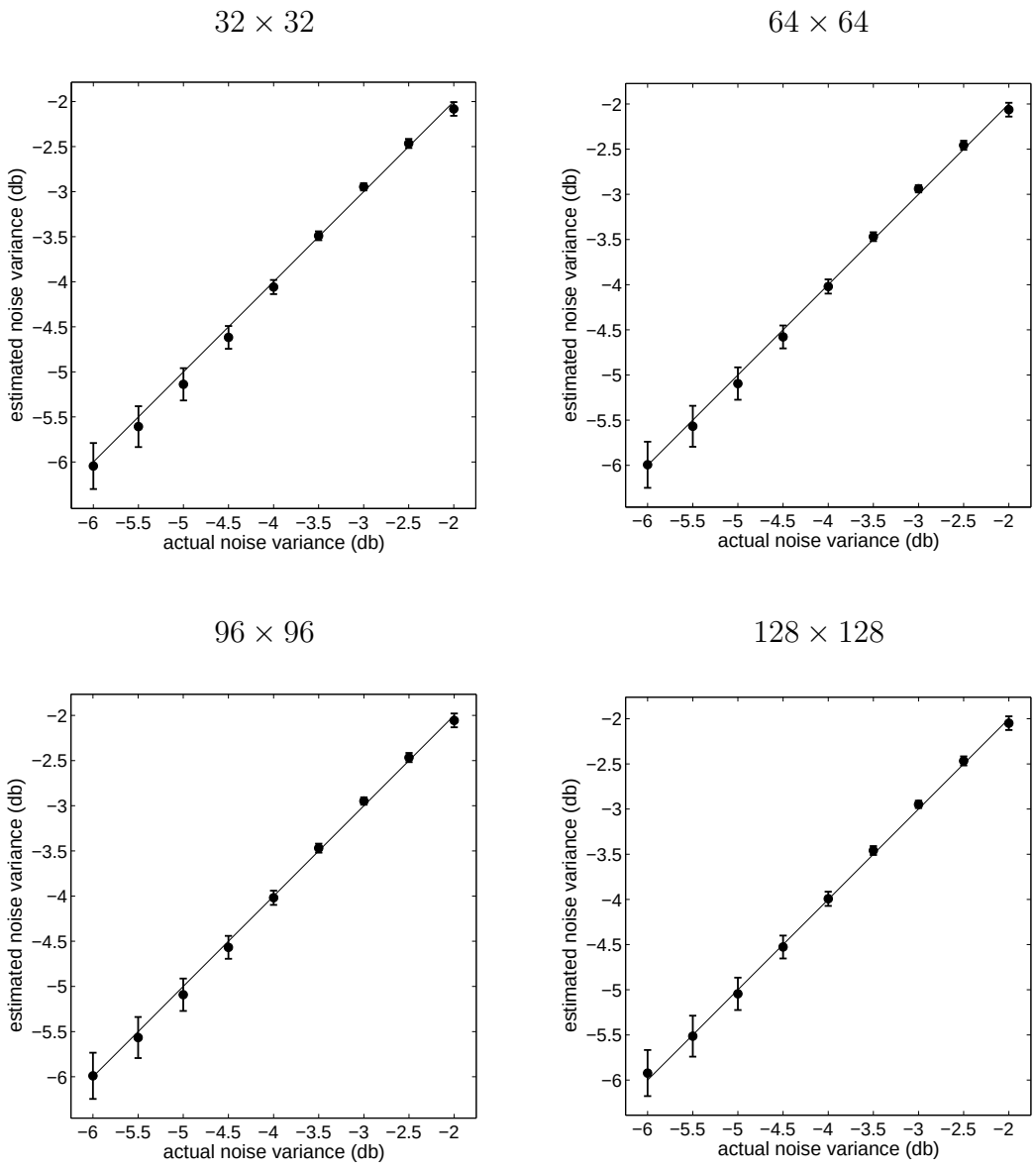

Figure 6.4: Shown are the average estimated noise variances and their standard deviations, plotted against the true values of the noise variances. Each data point corresponds to an average over 50 images using blocks of various sizes. See also Table 6.1.

White Gaussian noise with different variances was uniformly added to the color images in the database. For a given color image, the variance of the noise was the same for each channel. The results of blindly estimating the local noise variance from 50 color images (when the above heuristic is employed) are shown in Table 6.2, and summarized in Figure 6.5. The original and estimated noise variances are shown in decibels (db). Note that, for noise variances greater than -3db, the estimation is both accurate and precise, that is, the mean of the estimates is close to the actual value, and the standard deviation of the estimates is small. For noise variances smaller than -4db, however, the estimation fails. The reason for this poor performance is related to both the failure of our heuristic to accurately estimate the original local kurtosis, and to the errors in the other sample average estimates.

As in the experiments with single-channel images, we have noticed that due to numerical and sample average errors, a constant bias is introduced in the estimated noise variances. We have empirically found that a linear model suffices to approximate this bias well. We have employed the average estimates in the range between -3.5db and -1db (i.e., the range for which the discrimination of different noise levels is possible), to empirically estimate linear relationships between the estimated and true noise variances:

block size	linear relationship
32×32	$N = 1.2572\,\hat{N} - 0.91292$
64×64	$N = 1.2947\,\hat{N} - 0.87348$
96×96	$N = 1.2957\,\hat{N} - 0.85403$
128×128	$N = 1.2894\,\hat{N} - 0.82960$

These relationships were obtained using estimates from 50 color images different from the ones employed in the sensitivity experiments above. Note that the parameters of these linear relationships are practically independent of the window size used in the estimation.

6.3 Summary

We have presented a technique for blindly estimating local noise variance and showed how it can be applied to detect the presence of multiple levels of noise in an image. This technique assumes that the kurtosis of local patches of the original image is known, and that the noise is white, Gaussian, and statistically independent of the original image. The estimation method can be adapted to deal with different types of noise with known statistical properties (i.e., the kurtosis). We have shown the efficacy of measuring the local noise variance in natural images with locally added noise, and we have quantified the sensitivity of the estimation method.

32×32

actual	estimated (db)			
(db)	mean	std	min	max
-6.00	-3.771	0.510	-4.853	-2.822
-5.50	-3.773	0.510	-4.853	-2.823
-5.00	-3.776	0.506	-4.810	-2.825
-4.50	-3.757	0.475	-4.638	-2.822
-4.00	-3.651	0.384	-4.248	-2.823
-3.50	-3.405	0.260	-3.789	-2.774
-3.00	-3.019	0.157	-3.256	-2.593
-2.50	-2.531	0.101	-2.722	-2.246
-2.00	-2.003	0.089	-2.227	-1.774
-1.50	-1.483	0.082	-1.670	-1.283
-1.00	-0.982	0.057	-1.080	-0.841

64×64

actual	estimated (db)			
(db)	mean	std	min	max
-6.00	-3.717	0.562	-4.843	-2.475
-5.50	-3.720	0.563	-4.821	-2.479
-5.00	-3.725	0.563	-4.806	-2.473
-4.50	-3.723	0.552	-4.765	-2.472
-4.00	-3.640	0.475	-4.425	-2.463
-3.50	-3.432	0.346	-3.935	-2.421
-3.00	-3.075	0.208	-3.346	-2.395
-2.50	-2.589	0.108	-2.794	-2.291
-2.00	-2.042	0.075	-2.249	-1.826
-1.50	-1.485	0.082	-1.702	-1.277
-1.00	-0.990	0.085	-1.154	-0.771

96×96

actual	estimated (db)			
(db)	mean	std	min	max
-6.00	-3.691	0.594	-4.954	-2.346
-5.50	-3.695	0.594	-4.930	-2.352
-5.00	-3.699	0.592	-4.922	-2.354
-4.50	-3.698	0.582	-4.828	-2.352
-4.00	-3.618	0.501	-4.508	-2.348
-3.50	-3.420	0.367	-3.967	-2.337
-3.00	-3.076	0.225	-3.366	-2.338
-2.50	-2.598	0.116	-2.818	-2.294
-2.00	-2.052	0.080	-2.271	-1.882
-1.50	-1.493	0.085	-1.720	-1.320
-1.00	-0.988	0.091	-1.199	-0.788

128×128

actual	estimated (db)			
(db)	mean	std	min	max
-6.00	-3.675	0.607	-4.997	-2.314
-5.50	-3.678	0.606	-4.957	-2.314
-5.00	-3.684	0.606	-4.958	-2.317
-4.50	-3.681	0.595	-4.869	-2.318
-4.00	-3.605	0.512	-4.553	-2.314
-3.50	-3.412	0.376	-3.979	-2.305
-3.00	-3.075	0.231	-3.372	-2.326
-2.50	-2.604	0.119	-2.815	-2.300
-2.00	-2.060	0.081	-2.283	-1.905
-1.50	-1.497	0.084	-1.716	-1.344
-1.00	-0.983	0.091	-1.198	-0.787

Table 6.2: Shown are the statistics of the estimated noise variances (mean, standard deviation, and minimum and maximum values) as obtained for different block sizes. These statistics have been obtained from 50 natural images. See also Figure 6.5.

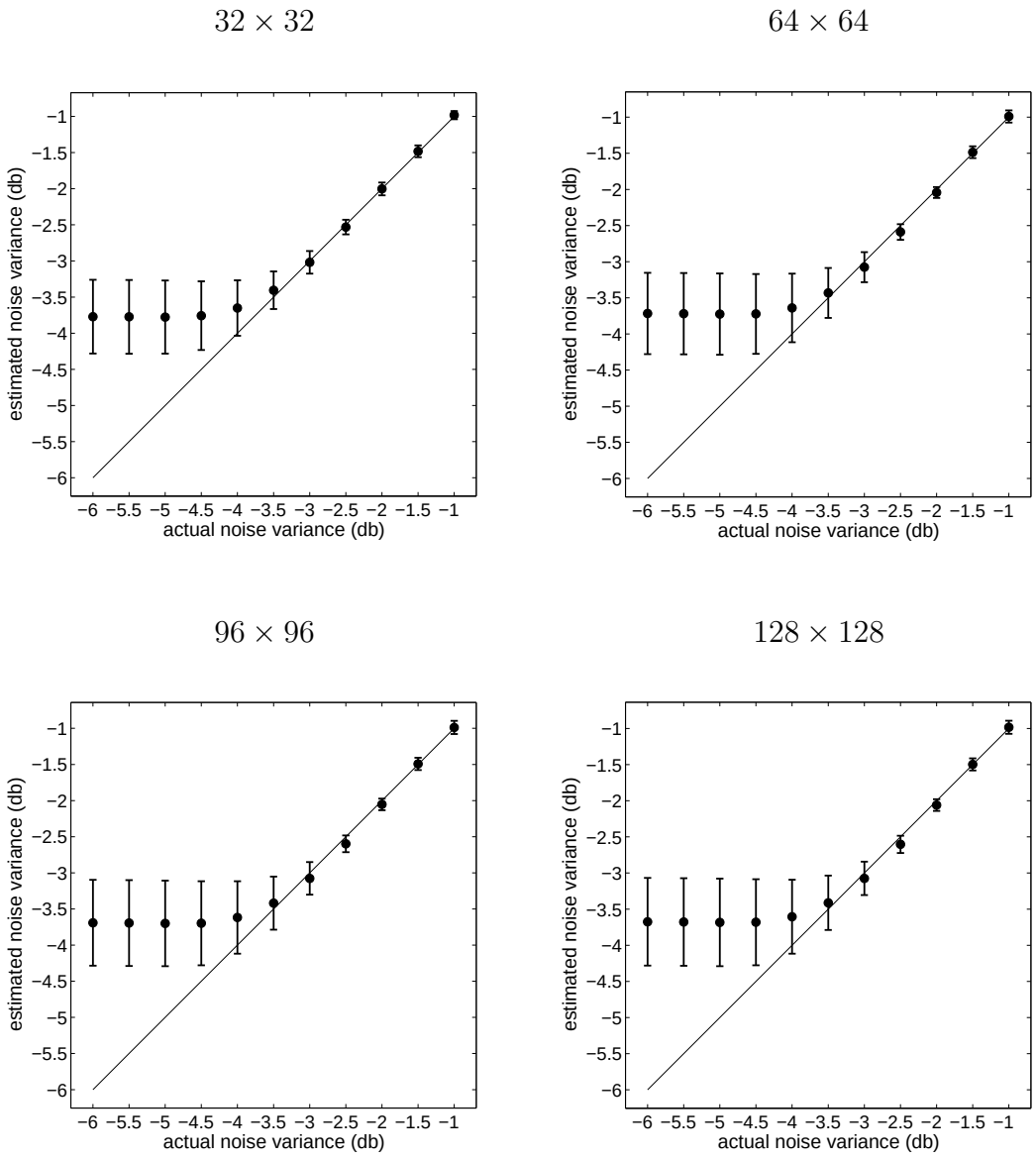

Figure 6.5: Shown are average estimated noise variances and their standard deviations, plotted against the true values of the noise variances. These estimates were obtained from the green channel of color images. The original local kurtosis was estimated from the average of the three color channels. Each data point corresponds to an average over 50 images using blocks of various sizes. See also Table 6.2.

Chapter 7

Conclusion

In the previous chapters, we have presented five techniques for detecting different forms of tampering in manipulated digital images: (1) traces of re-sampling [52] (e.g., scaling, rotation); (2) manipulations of color filter array interpolated images [53]; (3) double JPEG compression [51]; (4) duplicated regions [50]; and (5) inconsistent noise patterns [51]. Although these techniques are different, they work in a common framework under the assumption that tampering may alter some underlying statistical properties of natural images. For each technique, the general approach has been the same: first, statistical changes associated with specific types of tampering are identified and quantified, then detection methods are designed to estimate these changes and differentiate between tampered and unadulterated images.

When digitally compositing two, or more, images, it is often necessary to scale, rotate, or stretch the original images, or portions of them, in order to create a convincing match. These manipulations require re-sampling the original images onto a different sampling lattice. We have shown, Chapter 2, that re-sampling introduces specific correlations between neighboring image pixels. These correlations were quantified, and the expectation/maximization (EM) algorithm was employed to detect them in any portion of an image. This detection method can detect a broad range of re-sampling parameters, and is fairly robust to simple attacks. Experiments with lossy compressed images, however, have revealed a significant weakness in our technique: while reasonably robust on GIF and JPEG2000 images, our method is particularly sensitive to JPEG compression. This is because JPEG compression introduces a significant amount of band-pass noise, and specific artifacts that interfere with the detection of re-sampling correlations. We believe, however, that our technique may still prove useful for a number of different forensic applications: for example, a court of law may accept into evidence only JPEG images with minimal compression.

The re-sampling detection technique can also be employed to foil attacks on authentication watermarking schemes. Temporal and geometric distortions are a well-known class of attacks to which robust watermarking schemes are particularly sensitive. In audio signals, temporal scaling can occur from speed changes of a tape player, or can be the result of a sophisticated process designed to change the duration of sounds without altering the wavelengths of their spectra. In digital photographs, geometric distortions can be the result of manipulations with a photo-editing software. In video signals, both temporal and geometric distortions can occur when converting

between different encoding formats. These temporal and geometric distortions can be modeled with an affine transformation. More specifically, a watermarked signal is first affine transformed, then re-sampled onto a different sampling lattice to yield a new signal in which the embedded watermark becomes unrecognizable by the watermark detector. To foil this attack, the distortion parameters are first blindly estimated using our re-sampling detection technique[1], then the temporal or geometric distortions are inverted, and the embedded watermark recovered.

To capture color images, most digital cameras employ a single CCD or CMOS sensor in conjunction with a color filter array (CFA). At each pixel location, a single color sample (out of three) is recorded, and the missing color samples are interpolated from neighboring samples. We have shown, Chapter 3, that this process, referred to as CFA interpolation or demosaicking, introduces specific correlations between neighboring color samples, similar to those present in re-sampled images. A broad class of tampering techniques (e.g., re-touching, enhancing) are likely to destroy or significantly alter these correlations. As such, the presence, absence, or distortion of CFA interpolation correlations may represent evidence of authenticity or tampering. We have quantified these correlations with a simple linear model, and employed the expectation/maximization (EM) algorithm to reveal their presence, or absence, in any portion of a CFA interpolated color image. We have tested our technique on images CFA interpolated with seven different schemes, and found that we can detect traces of tampering even in lossy compressed images. Although our method is quite robust to simple counter attacks, we have found a vulnerability. If a tampered image is re-sampled onto a virtual CFA, then re-interpolated, the traces of tampering become undetectable. This attack, however, requires knowledge of the camera's CFA pattern and interpolation scheme, and is not likely to be available to an average user. Another potential issue with this detection method is the release of a new type of CMOS sensor, the Foveon X3, that can simultaneously sample the three color channels at each pixel location. Images from cameras equipped with this sensor will no longer contain the CFA interpolation correlations employed to detect tampering. We believe, however, that our detection technique may still be useful given that traditional single sensor cameras are likely to remain cheaper and more available to casual users.

A photo-editing software package (e.g., *Adobe Photoshop*®) is usually employed to manipulate digital images. An image is first loaded into the photo-editing software, some manipulations are performed, then it is re-saved. If the original and tampered images are stored in JPEG format, the default option for most digital cameras, then the tampered image has been double compressed. We have shown, Chapter 4, that when an image is double JPEG compressed specific correlations are introduced between discrete cosine transform (DCT) coefficients of image blocks. We have quantified these correlations, and devised an algorithm that can distinguish between single and double JPEG compressed images. Although there are certain theoretical limitations that make double compressed images undetectable in some cases, we have tested our detection method on a large number of images and found that double JPEG compression is detectable for a broad range of quality factors. We have also found that our detection method is vulnerable to attack: if a tampered JPEG image is cropped prior to re-saving, the correlations described in Chapter 4 are no longer

[1]It is possible that more than one set of distortion parameters are obtained from our detection method, e.g., image rotations by angles θ° and $-\theta^\circ$ are indistinguishable. In this case, the signal must be inverted with each possible set of distortion parameters, and the watermark detector must be applied to each inverted signal.

introduced. It is possible that other artifacts may be present, and this will be an interesting problem for further research. Note that a double JPEG compressed image is not necessarily a forgery: for example, a user may re-save JPEG images with a lower quality to save storage space. The authenticity of a double JPEG compressed image, however, is questionable and further analysis may be required.

When trying to conceal a person or object in a digital image, a common manipulation is to copy a portion of the same image, then paste it over the desired region. If the splicing is imperceptible little concern is given to the fact that identical (or virtually identical) regions are present. An exhaustive approach to detecting duplicated image regions is computationally prohibitive — it has an exponential complexity in the number of image pixels. We have proposed an efficient technique, Chapter 5, that can detect and localize duplicated regions in an image. The basic approach works by first applying a principal component analysis (PCA) on small, fixed-size image blocks to yield a reduced-dimensional representation robust to noise and lossy compression. Duplicated regions are detected by lexicographically sorting the transformed image blocks, then examining pairs of neighboring blocks. We have shown the efficacy of this technique on credible forgeries, and analyzed its sensitivity to simple attacks and lossy compression. We have found that duplicated regions of relatively small size (e.g., 32×32 pixels) are harder to detect, particularly in the presence of noise. Note that this method requires human intervention to validate the potential duplicated regions returned by our algorithm. This is because fairly large regions with little variation (e.g., sky, asphalt roads) present in natural images tend to yield false positives, i.e., regions that are similar, but not duplicated. In our experiments, we have found that our algorithm yields a relatively low average number of false positives, i.e., less than one percent. Our technique can also be employed to detect the presence of duplicated regions in two, or more, different images, as might occur when a composite image is created by splicing together portions of several images. The principal component analysis is applied to fixed-size blocks from all images, then the algorithm proceeds as in the single image case.

Digital cameras introduce an inherent amount of noise uniformly spread across an image. When creating digital forgeries, it is common to add a small amount of additive noise to conceal traces of tampering, e.g., to obscure details in the background or at a splice boundary. As a result, tampered images may contain inconsistent noise patterns, i.e., significant variations in the local noise variance. Measuring the noise variance in the absence of the original, noiseless image is a non-trivial, ill-defined problem. We have employed a *blind* signal to noise ratio (SNR) estimator, Chapter 6, to locally estimate the noise variance in an image and reveal the presence of inconsistent noise levels. This estimator works under the assumption that the image and the noise are statistically independent, and requires knowledge of the kurtosis values of local image patches, and the kurtosis of the noise. While the kurtosis of the noise may be known (e.g., it is equal to 3 for white Gaussian noise) or easily estimated, reliably estimating the kurtosis of the original image patches remains an open problem. Assuming, for gray-scale images, that the original values of the local kurtosis are known, and, for color images, employing a simple heuristic to estimate them, we have illustrated how blind estimation of the local noise variance can reveal the presence of inconsistent noise levels (a sign of tampering) in images with locally added noise. We have found, however, that due to numerical errors, and to errors in the sample average estimates of the kurtosis, a bias

is introduced in the estimation of the local noise variance. This bias was quantified, then removed from the estimation. Sensitivity experiments have shown that, in general, the estimation is fairly accurate. Not unexpectedly, we have noticed that due to numerical errors the estimation becomes less accurate for very small noise variances.

As with any authentication schemes, our statistical techniques may be vulnerable to counter-attack. Extensive experiments have shown that our algorithms are reasonably robust to simple attacks, e.g., lossy compression schemes, additive noise, and non-linear luminance transformations. We have, however, identified several more sophisticated attacks to which some of our detection schemes are vulnerable. For example, re-sampling detection is particularly sensitive to lossy JPEG compression, the correlations specific to color filter array interpolation can be simulated if the color pattern and interpolation scheme are known, and double quantization artifacts are no longer introduced in a double JPEG compressed image, if the image is cropped prior to the second compression. In a forensic context, it is important that the development of authentication tools and the analysis of their limitations go in parallel. A full analysis of these, and other, potential vulnerabilities is beyond the scope of this thesis, and represents an interesting direction for future research.

Most techniques proposed in this thesis were targeted towards detecting tampering in uncompressed or lossless compressed images. We have noticed that lossy compression schemes (e.g., JPEG) introduce specific artifacts that interfere with some of our statistical methods. Since lossy compressed images are more widespread than their uncompressed counterparts (a significant proportion of digital cameras store images only in JPEG format) an important direction for future research would be to design detection techniques targeted towards lossy compressed images.

Another interesting future research topic would be to investigate statistical techniques for detecting tampering in digitally scanned images. For example, color images captured with single sensor cameras contain correlations specific to color filter array interpolation that can be employed for image authentication. It may be worth investigating if similar correlations are introduced by digital scanners, and if digitally scanned images and digital camera images have different statistical properties. This may also help foil the analog to digital attack in which a digital image is printed, then scanned in order to destroy embedded watermarks, or to conceal traces of tampering.

With the advent of low-cost, powerful computers and sophisticated editing tools even casual users can manipulate digital video with relative ease. The resulting tampered video sequences can be very hard to distinguish from authentic footage. It is likely, however, that tampering might alter some underlying statistical properties of natural video sequences. A future research topic would be to investigate what statistical changes occur when tampering with video sequences, and to design detection techniques that quantify and estimate these changes. A potential issue is that most digital video sequences are encoded with lossy compression techniques (e.g., MPEG), which is likely to introduce artifacts (similar to those introduced by JPEG compression in images) that may conceal traces of tampering. We hope that the significant amounts of data contained in video sequences might offset this potential difficulty.

The proposed statistical image authentication techniques work in the absence of any embedded watermarks or signatures. They are not, however, intended to replace the existing image authentication techniques, notably digital watermarking, but rather to provide a complementary approach.

We believe the development of statistical tools is particularly important to help contend with cases where existing techniques are not applicable, or may prove too costly. In addition, the statistical models employed to detect tampering may have applications in other fields, and their development may help provide a better understanding of the statistical properties of natural images.

We believe that many complementary statistical techniques such as those presented here [52, 51, 50, 53], and those that others are developing (e.g., [18, 40, 24, 39]), will be needed to reliably expose digital forgeries. There is little doubt that even with the development of a suite of detection techniques, more sophisticated tampering techniques will emerge, which in turn will lead to the development of more elaborate detection tools, and so on. Our hope, however, is that our detection tools would make the creation of undetectable forgeries an increasingly complex and difficult task.

Appendix A

Expectation-Maximization Algorithm

We present a short tutorial of the Expectation-Maximization (EM) algorithm [14], a general technique for finding maximum-likelihood parameter estimates of probabilistic models. We also present a simple example of its application to a mixture of two linear models (for a more detailed tutorial and examples see [6]).

A.1 Preliminaries and General Approach of EM

Consider a density function $p(\vec{x} \mid \Theta)$ that depends on a set of parameters Θ, e.g., if $p(\vec{x} \mid \Theta)$ were a mixture (sum) of Gaussians, then Θ would be the set of their means and covariances. Let $X = \{\vec{x}_1, \ldots, \vec{x}_N\}$ be a data set consisting of N vectors, independent and identically distributed (i.i.d.) with distribution p. The joint density of the samples is given by:

$$p(X \mid \Theta) = \prod_{i=1}^{N} p(\vec{x}_i \mid \Theta) = \mathcal{L}(\Theta \mid X), \tag{A.1}$$

where $\mathcal{L}(\Theta \mid X)$, as a function of Θ with fixed data X, is called the likelihood of the parameters given the data, or the likelihood function. Maximum likelihood estimation amounts to finding $\Theta^\star$ for which $\mathcal{L}(\Theta \mid X)$ is maximized, i.e.:

$$\Theta^\star = \arg\max_{\Theta} \mathcal{L}(\Theta \mid X). \tag{A.2}$$

In practice, it is often analytically easier to maximize the log-likelihood, $\log(\mathcal{L}(\Theta \mid X))$, instead, e.g., when $p(\vec{x} \mid \Theta)$ is an exponential distribution such as a Gaussian or Laplacian.

In the context of the EM algorithm, the data is assumed to be incomplete or to have missing values. This is the case when the observation process has limitations, or when the optimization of the likelihood function is intractable and can be simplified by assuming the existence of unknown data or missing variables (also known as hidden, or latent variables). A *complete* data set $Z = (X, Y)$ is assumed to exist, where X, the observed data, is called the *incomplete data*, and Y

denotes the *missing* parameters or data. A joint density function is also assumed to exist, or is specified:

$$p\left(\vec{z}\,|\,\Theta\right) = p\left(\vec{x},\vec{y}\,|\,\Theta\right) = p\left(\vec{y}\,|\,\vec{x},\Theta\right)\; p\left(\vec{x}\,|\,\Theta\right). \tag{A.3}$$

This joint distribution is often constructed from the marginal density function $p\left(\vec{x}\,|\,\Theta\right)$ of the incomplete (observed) data, and the assumption of hidden variables, as in the case of mixture densities. A new likelihood function, the complete data likelihood, is defined from the joint density function:

$$\mathcal{L}\left(\Theta\,|\,Z\right) = \mathcal{L}\left(\Theta\,|\,X,Y\right) = p\left(X,Y\,|\,\Theta\right). \tag{A.4}$$

The goal of the EM algorithm is to maximize the incomplete data log-likelihood function $\log\left(\mathcal{L}\left(\Theta\,|\,X\right)\right)$. Given $\Theta^{(n)}$, a current choice of parameter estimates, the incomplete data log-likelihood takes the form:

$$\log\left(\mathcal{L}\left(\Theta^{(n)}\,|\,X\right)\right) = \log p\left(X\,|\,\Theta^{(n)}\right) = \log\prod_{i=1}^{N} p\left(\vec{x}_i\,|\,\Theta^{(n)}\right) = \sum_{i=1}^{N}\log p\left(\vec{x}_i\,|\,\Theta^{(n)}\right). \tag{A.5}$$

Note that the marginal density on the incomplete data, $p\left(\vec{x}\,|\,\Theta\right)$, can be obtained as the expected value of the joint density of the complete data, $p\left(\vec{x},\vec{y}\,|\,\Theta\right)$, Equation (A.3). This expectation is computed with respect to the missing data Y given the observed data X:

$$p\left(\vec{x}\,|\,\Theta\right) = \int_{Y\in\Upsilon} p\left(\vec{x}\,|\,\Theta\right) p\left(Y\,|\,\vec{x},\Theta\right) dY = \int_{Y\in\Upsilon} p\left(\vec{x},Y\,|\,\Theta\right) dY, \tag{A.6}$$

where Υ denotes the space of values that Y can take on. Substituting Equation (A.6) into Equation (A.5) yields the following expression for the incomplete data log-likelihood:

$$\log\left(\mathcal{L}\left(\Theta^{(n)}\,|\,X\right)\right) = \sum_{i=1}^{N}\log\int_{Y\in\Upsilon} p\left(\vec{x}_i,Y\,|\,\Theta^{(n)}\right) dY. \tag{A.7}$$

We would like next to find new parameter estimates $\Theta^{(n+1)}$ that increase the log-likelihood, i.e., maximize the difference between the original and new log-likelihoods:

$$\Theta^{(n+1)} = \arg\max_{\Theta} Q(\Theta,\Theta^{(n)}) \tag{A.8}$$

where:

$$Q(\Theta,\Theta^{(n)}) = \log\left(\mathcal{L}\left(\Theta\,|\,X\right)\right) - \log\left(\mathcal{L}\left(\Theta^{(n)}\,|\,X\right)\right). \tag{A.9}$$

In practice, however, it is often easier to maximize a lower bound of $Q(\Theta,\Theta^{(n)})$ instead. This

lower bound is obtained using some algebraic manipulations:

$$\begin{aligned}
Q(\Theta, \Theta^{(n)}) &= \sum_{i=1}^{N} \log \int_{Y \in \Upsilon} p\left(\vec{x}_i, Y \mid \Theta\right) dY - \sum_{i=1}^{N} \log \int_{Y' \in \Upsilon} p\left(\vec{x}_i, Y' \mid \Theta^{(n)}\right) dY' \\
&= \sum_{i=1}^{N} \log \frac{\int_{Y \in \Upsilon} p\left(\vec{x}_i, Y \mid \Theta\right) dY}{\int_{Y' \in \Upsilon} p\left(\vec{x}_i, Y' \mid \Theta^{(n)}\right) dY'} \\
&= \sum_{i=1}^{N} \log \int_{Y \in \Upsilon} \frac{p\left(\vec{x}_i, Y \mid \Theta\right) dY}{\int_{Y' \in \Upsilon} p\left(\vec{x}_i, Y' \mid \Theta^{(n)}\right) dY'} \\
&= \sum_{i=1}^{N} \log \int_{Y \in \Upsilon} \frac{p\left(\vec{x}_i, Y \mid \Theta^{(n)}\right)}{\int_{Y' \in \Upsilon} p\left(\vec{x}_i, Y' \mid \Theta^{(n)}\right) dY'} \frac{p\left(\vec{x}_i, Y \mid \Theta\right)}{p\left(\vec{x}_i, Y \mid \Theta^{(n)}\right)} dY \\
&= \sum_{i=1}^{N} \log \int_{Y \in \Upsilon} p\left(Y \mid \vec{x}_i, \Theta^{(n)}\right) \frac{p\left(\vec{x}_i, Y \mid \Theta\right)}{p\left(\vec{x}_i, Y \mid \Theta^{(n)}\right)} dY.
\end{aligned} \tag{A.10}$$

The conditional probability rule and Equation (A.6) were employed in the last step of Equation (A.10):

$$p\left(Y \mid \vec{x}_i, \Theta\right) = \frac{p\left(\vec{x}_i, Y \mid \Theta\right)}{p\left(\vec{x}_i \mid \Theta\right)} = \frac{p\left(\vec{x}_i, Y \mid \Theta\right)}{\int_{Y' \in \Upsilon} p\left(\vec{x}_i, Y' \mid \Theta^{(n)}\right) dY'}. \tag{A.11}$$

Using Jensen's inequality[1] (the $\log$ is a concave function) in Equation (A.10) yields the desired lower bound:

$$\begin{aligned}
Q(\Theta, \Theta^{(n)}) &= \sum_{i=1}^{N} \log \int_{Y \in \Upsilon} p\left(Y \mid \vec{x}_i, \Theta^{(n)}\right) \frac{p\left(\vec{x}_i, Y \mid \Theta\right)}{p\left(\vec{x}_i, Y \mid \Theta^{(n)}\right)} dY \\
&\geq \sum_{i=1}^{N} \int_{Y \in \Upsilon} p\left(Y \mid \vec{x}_i, \Theta^{(n)}\right) \log \frac{p\left(\vec{x}_i, Y \mid \Theta\right)}{p\left(\vec{x}_i, Y \mid \Theta^{(n)}\right)} dY = L(\Theta, \Theta^{(n)}).
\end{aligned} \tag{A.12}$$

To understand how the lower bound in Equation (A.12) is employed, consider an arbitrary function f, and let g be a lower bound of f, i.e., $f(x) \geq g(x), \forall x$. If there exists x_0 such that $f(x_0) = g(x_0)$, and $x^\star$ such that $g(x^\star) > g(x_0)$, then $f(x^\star) > f(x_0)$ — this is particularly true when $x^\star = \arg\max_x g(x)$. In other words, if by moving from x_0 to $x^\star$ the lower bound g is increased, then the function f is also increased. Note that the lower bound in Equation (A.12) is constructed such that $L(\Theta^{(n)}, \Theta^{(n)}) = Q(\Theta^{(n)}, \Theta^{(n)}) = 0$. As a result, a new choice of parameters $\Theta^{(n+1)}$ that improves the lower bound will also improve $Q(\Theta, \Theta^{(n)})$, Equation (A.9), which is

[1] Jensen's inequality states that given f a real continuous concave function then:

$$\frac{\sum_{i=1}^{n} f(x_i)}{n} \leq f\left(\frac{\sum_{i=1}^{n} x_i}{n}\right).$$

This inequality can be generalized to infi nite or continuous sums, i.e., expectations or integrals.

equivalent to increasing the incomplete data log-likelihood:

$$L(\Theta^{(n+1)}, \Theta^{(n)}) > L(\Theta^{(n)}, \Theta^{(n)}) \quad \Rightarrow \quad Q(\Theta^{(n+1)}, \Theta^{(n)}) > Q(\Theta^{(n)}, \Theta^{(n)}) \quad \Leftrightarrow$$
$$\log\left(\mathcal{L}\left(\Theta^{(n+1)} \mid X\right)\right) > \log\left(\mathcal{L}\left(\Theta^{(n)} \mid X\right)\right) \quad \text{(A.13)}$$

Note also that in the definition of $L(\Theta, \Theta^{(n)})$, Equation (A.12), the denominator of the $\log$ does not depend on Θ, and can be dropped when maximizing the lower bound with respect to Θ.

We are finally ready to present the general approach of the EM algorithm. First, initial parameter estimates $\Theta^{(0)}$ are chosen[2]. The algorithm then proceeds in iterations, at each iteration two steps being executed:

1. During the *E-step*[3] the expected value of the complete data log-likelihood, Equation (A.4), is computed with respect to the unknown data Y given the observed data X and a current setting of the parameter estimates $\Theta^{(n)}$:

$$\begin{aligned}\mathcal{E}\left\{\log\left(\mathcal{L}\left(Z \mid \Theta\right)\right) \mid X, \Theta^{(n)}\right\} &= \mathcal{E}\left\{\log\, p\left(X, Y \mid \Theta\right) \mid X, \Theta^{(n)}\right\} \\ &= \int_{Y \in \Upsilon} \log\left[p\left(X, Y \mid \Theta\right)\right]\, p\left(Y \mid X, \Theta^{(n)}\right) dY \\ &= \int_{Y \in \Upsilon} \left[\sum_{i=1}^{N} \log p\left(\vec{x}_i, Y \mid \Theta\right)\right]\, p\left(Y \mid X, \Theta^{(n)}\right) dY.\end{aligned} \quad \text{(A.14)}$$

   Note that the expectation in Equation (A.14) is equivalent to the lower bound in Equation (A.12) with the denominator of the $\log$ dropped. Note also that the expectation of the complete data log-likelihood is a deterministic function of Θ that depends on the observed data, X, and on the current parameter setting, $\Theta^{(n)}$.

2. During the *M-step* the expectation computed during the E-step is maximized with respect to Θ to yield new parameter estimates:

$$\begin{aligned}\Theta^{(n+1)} &= \arg\, \max_{\Theta} \mathcal{E}\left\{\log\left(\mathcal{L}\left(Z \mid \Theta\right)\right) \mid X, \Theta^{(n)}\right\} \\ &= \arg\, \max_{\Theta} \int_{Y \in \Upsilon} \left[\sum_{i=1}^{N} \log p\left(\vec{x}_i, Y \mid \Theta\right)\right]\, p\left(Y \mid X, \Theta^{(n)}\right) dY.\end{aligned} \quad \text{(A.15)}$$

The two steps are repeated until convergence. Each iteration is guaranteed to increase the incomplete log-likelihood. The algorithm is proved to converge to a local maximum of the likelihood function, Equation (A.1) — see [62] for details on the rate of convergence of the EM algorithm.

When presented as above, in its most general form, the EM algorithm does not provide exact implementation details. The details of the E- and M-steps are dependent on the particular application the algorithm is designed for. We present next a simple example where the EM algorithm is applied to a mixture of two linear models.

[2]The initial parameter estimates are often randomly chosen. When available, prior information can be employed for a more "educated" guess.

[3]Speaking of an expectation (E) step is somewhat a misnomer. What is really computed in the first step are the data-dependent terms of the lower bound in Equation (A.12), under the assumption of existence of missing parameters. Once computed, these terms fully determine the lower bound which can be maximized in the second step.

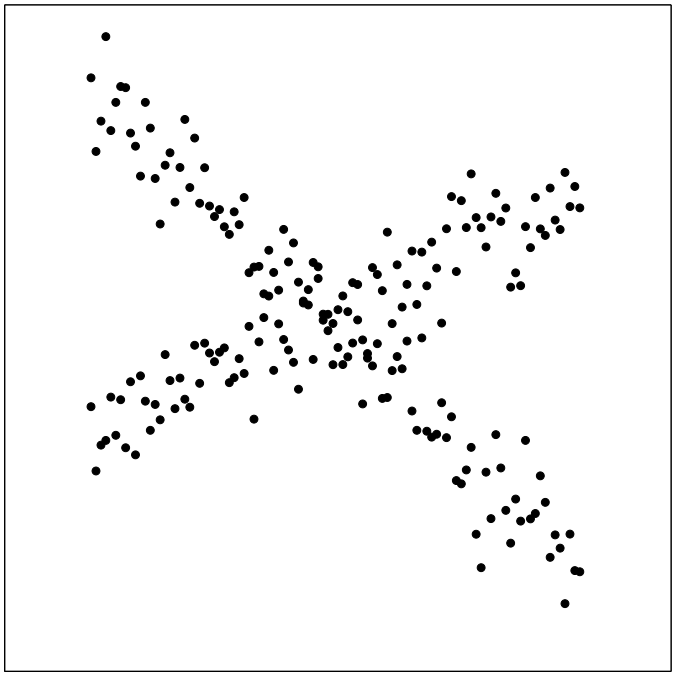

Figure A.1: Planar points generated from two linear models corrupted with white Gaussian noise.

A.2 Mixture of Two Linear Models

The EM algorithm provides an elegant solution for mixture models problems, as it can simultaneously segment and fit data generated from multiple parametric models. Consider, for example, a set of 2-D data points $X = \{(x_i, y_i) \mid i = 1, \ldots, N\}$, Figure A.1, each of them generated from one of two linear models:

$$m_1 : \; y_i = a_1 x_i + b_1 + n_1(i) \quad \text{or} \quad m_2 : \; y_i = a_2 x_i + b_2 + n_2(i), \tag{A.16}$$

where a_1, b_1, a_2, b_2 are the model parameters, and $n_1(i)$, $n_2(i)$ denote i.i.d. samples drawn from a Gaussian distribution with zero mean and σ^2 variance. In the context of the EM algorithm, the observed data is represented by the set of points X, and the parameters are the slopes and the intercepts of the linear models: $\Theta = \{a_1, b_1, a_2, b_2\}$. The missing data are represented by a set of N random variables, one for each data point, that can take one of two values: m_1, if the corresponding data point is generated by the first model, or m_2, if the corresponding data point is generated by the second model.

To maximize the observed data likelihood $\mathcal{L}(X \mid \Theta)$, Equation (A.1), we start with randomly chosen parameters $\Theta^{(0)} = \{a_1^{(0)}, b_1^{(0)}, a_2^{(0)}, b_2^{(0)}\}$, then at each iteration we maximize the lower bound of the observed data log-likelihood:

$$J(\Theta, \Theta^{(n)}) = \sum_{j=1}^{2} \sum_{i=1}^{N} \log \left[p\left((x_i, y_i), m_j \mid \Theta\right) \right] \, p\left(m_j \mid (x_i, y_i), \Theta^{(n)}\right). \tag{A.17}$$

The above lower bound can be obtained from the more general form in Equation (A.14) under the assumption that the points in X are independently generated and the missing (unobserved) random variables are mutually independent. In Equation (A.17), $p\left((x_i, y_i), m_j \mid \Theta\right)$ is the probability that the i^{th} point was generated by model m_j and that its observed coordinates were (x_i, y_i), given the model parameters Θ. Likewise, $p\left(m_j \mid (x_i, y_i), \Theta^{(n)}\right)$ is the probability that i^{th} point was generated by model m_j, given that its observed coordinates were (x_i, y_i) and the model parameters are $\Theta^{(n)}$. These probabilities need to be evaluated in order to maximize the lower bound $J(\Theta, \Theta^{(n)})$ as a function of Θ. To compute $p\left((x_i, y_i), m_j \mid \Theta\right)$, we start by applying the conditional probability rule:

$$p\left((x_i, y_i), m_j \mid \Theta\right) = p\left((x_i, y_i) \mid m_j, \Theta\right) p\left(m_j \mid \Theta\right), \tag{A.18}$$

where $p\left(m_j \mid \Theta\right)$ represents the prior probability of the i^{th} point being generated by model m_j. Since the points in X are generated from linear models corrupted by additive Gaussian noise, $p\left((x_i, y_i) \mid m_j, \Theta\right)$ is given by:

$$p\left((x_i, y_i) \mid m_j, \Theta\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - a_j x_i - b_j)^2}{2\sigma^2}\right), \qquad j = 1, 2 \quad i = 1, \ldots, N. \tag{A.19}$$

We will assume next that each point is equally likely to be generated by each model, i.e., the prior model probabilities are equal:

$$p\left(m_1 \mid \Theta\right) = p\left(m_2 \mid \Theta\right) = \frac{1}{2}. \tag{A.20}$$

The $\log$ term in Equation (A.17) then takes the form:

$$\log\left[p\left((x_i, y_i), m_j \mid \Theta\right)\right] = -\frac{1}{2\sigma}\left(y_i - a_j x_i - b_j\right)^2 - \log 2\sigma\sqrt{2\pi} \qquad j = 1, 2 \quad i = 1, \ldots, N. \tag{A.21}$$

To compute $p\left(m_j \mid (x_i, y_i), \Theta^{(n)}\right)$, the Bayes' rule and Equations (A.19-A.20) are employed:

$$\begin{aligned} p\left(m_j \mid (x_i, y_i), \Theta^{(n)}\right) &= \frac{p\left((x_i, y_i) \mid m_j, \Theta^{(n)}\right) p\left(m_j \mid \Theta^{(n)}\right)}{\sum_{k=1}^{2} p\left((x_i, y_i) \mid m_k, \Theta^{(n)}\right) p\left(m_k \mid \Theta^{(n)}\right)} \\ &= \frac{\exp\left(-\frac{1}{2\sigma^2}\left(y_i - a_j^{(n)} x_i - b_j^{(n)}\right)^2\right)}{\exp\left(-\frac{1}{2\sigma^2}\left(y_i - a_1^{(n)} x_i - b_1^{(n)}\right)^2\right) + \exp\left(-\frac{1}{2\sigma^2}\left(y_i - a_2^{(n)} x_i - b_2^{(n)}\right)^2\right)} \\ & \qquad j = 1, 2 \quad i = 1, \ldots, N. \end{aligned} \tag{A.22}$$

Substituting Equations (A.21-A.22) in Equation (A.17) and dropping the terms that do not depend on the parameters $\Theta = \{a_1, b_1, a_2, b_2\}$, yields the desired lower bound as a function of the parameters:

$$J(\Theta, \Theta^{(n)}) = -\sum_{j=1}^{2} \sum_{i=1}^{N} w(i, j)(y_i - a_j x_i - b_j)^2, \tag{A.23}$$

where $w(i,j) = p\left(m_j \mid (x_i, y_i), \Theta^{(n)}\right)$, Equation (A.22). This quadratic lower bound is maximized by first setting its partial derivatives equal to zero to yield a linear system of equations:

$$\begin{aligned} \frac{\partial J}{\partial a_j} &= 0 \\ \frac{\partial J}{\partial b_j} &= 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} 2\sum_{i=1}^{N} w(i,j)\, x_i \left(y_i - a_j x_i - b_j\right) &= 0 \\ 2\sum_{i=1}^{N} w(i,j) \left(y_i - a_j x_i - b_j\right) &= 0 \end{aligned} \quad \Rightarrow \tag{A.24}$$

$$\begin{aligned} a_j \left(\sum_{i=1}^{N} w(i,j)\, x_i^2\right) + b_j \left(\sum_{i=1}^{N} w(i,j)\, x_i\right) &= \sum_{i=1}^{N} w(i,j)\, x_i y_i \\ a_j \left(\sum_{i=1}^{N} w(i,j)\, x_i\right) + b_j \left(\sum_{i=1}^{N} w(i,j)\right) &= \sum_{i=1}^{N} w(i,j)\, y_i \end{aligned} \qquad j = 1, 2\,. \tag{A.25}$$

The above linear system of equations is then solved to obtain new parameter estimates:

$$\begin{bmatrix} a_j^{(n+1)} \\ b_j^{(n+1)} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} w(i,j)\, x_i^2 & \sum_{i=1}^{N} w(i,j)\, x_i \\ \sum_{i=1}^{N} w(i,j)\, x_i & \sum_{i=1}^{N} w(i,j) \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{N} w(i,j)\, x_i y_i \\ \sum_{i=1}^{N} w(i,j)\, y_i \end{bmatrix} \quad j = 1, 2\,. \tag{A.26}$$

Note that the lower bound $J(\Theta, \Theta^{(n)})$, Equation (A.23), is the weighted square error between the data points and the linear models. The weights are given by the probabilities of each point being generated by one of the two models. As a result, maximizing $J(\Theta, \Theta^{(n)})$ is equivalent to weighted least squares estimation.

To summarize, the EM algorithm proceeds as follows:

1. During the E-step, using the current parameter estimates, we compute the probabilities of each data point of belonging to one of the two models, Equation (A.22).

2. During the M-step, weighted least squares estimation is employed to compute new model parameters using the previously computed probabilities as weights, Equation (A.26).

The two steps are repeated until convergence. In practice, the EM algorithm converges quickly, but it is susceptible to local optima. This problem can be alleviated by running the EM algorithm multiple times with different random initial conditions.

Shown in Figure A.2 are the estimates of the linear models at each iteration of the EM algorithm using the data points in Figure A.1. The initial parameters were randomly chosen. Note that, in this example, the EM algorithm converges in only six iterations.

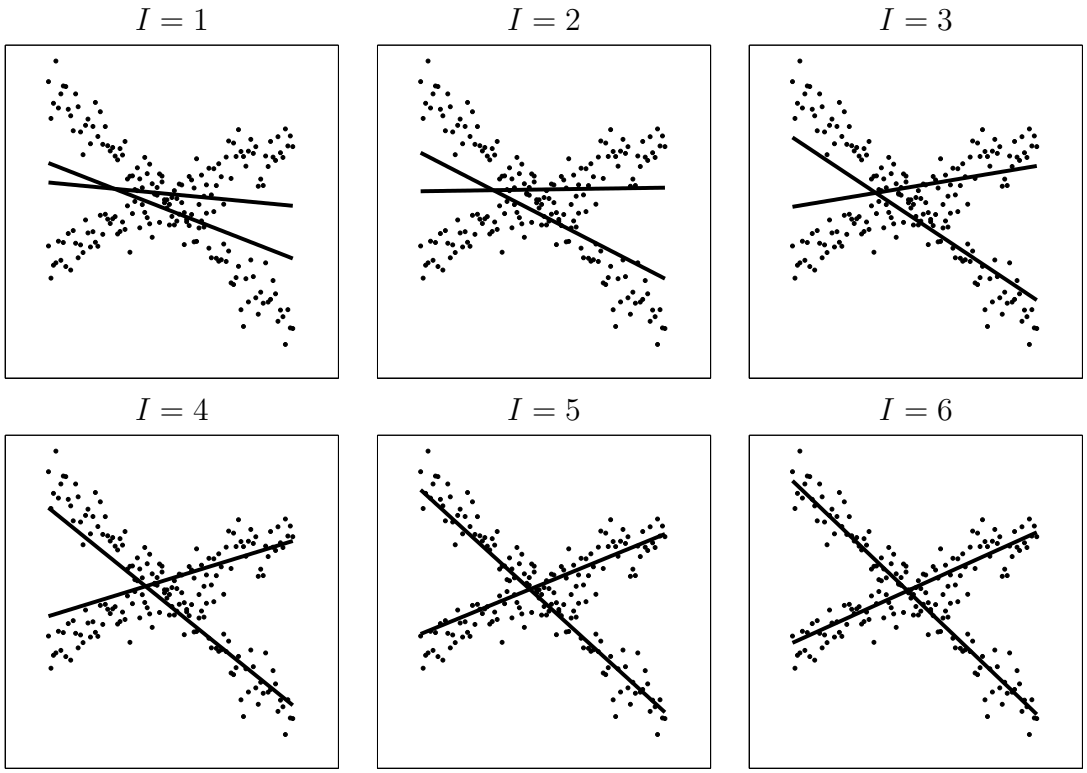

Figure A.2: Model estimates at each iteration of the EM algorithm for the data points in Figure A.1. Note that, in this example, convergence is achieved after six iterations.

Bibliography

[1] Digital compression and coding of continuous-tone still images, part 1: Requirements and guidelines. ISO/IEC JTC1 Draft International Standard 10918-1, 1991.

[2] Studio encoding parameters of digital television for standard $4:3$ and wide-screen $16:9$ aspect ratios. ITU-R Recommendation BT.601-5, 1995.

[3] Information technology — JPEG 2000 image coding system — part 1: Core coding system. ISO/IEC 15444-1, 2000.

[4] B. E. Bayer. Color imaging array. US Patent, 3971065, 1976.

[5] S. Bhattacharjee and M. Kutter. Compression-tolerant image authentication. In *IEEE International Conference on Image Processing*, volume 1, pages 435–439, 1998.

[6] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, International Computer Science Institute, Berkeley, CA, April 1998.

[7] P. Blythe and J. Fridrich. Secure digital camera. In *Digital Forensic Research Workshop*, Baltimore, Maryland, August 2004.

[8] M. U. Celik, G. Sharma, E. Saber, and A. M. Tekalp. Hierarchical watermarking for secure image authentication with localization. *IEEE Transactions on Image Processing*, 11(6):585–595, June 2002.

[9] E. Chang, S. Cheung, and D. Y. Pan. Color filter array recovery using a threshold-based variable number of gradients. In N. Sampat and T. Yeh, editors, *Sensors, Cameras, and Applications for Digital Photography, Proceedings of the SPIE*, volume 3650, pages 36–43, March 1999.

[10] D. R. Cok. Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal. US Patent, 4642678, 1987.

[11] I. J. Cox, M. L. Miller, and J. A. Bloom. *Digital Watermarking*. Morgan Kaufmann Publishers, 2002.

[12] I. J. Cox, M. L. Miller, J. M. G. Linnartz, and T. Kalker. *Digital Signal Processing for Multimedia Systems*, chapter A Review of Watermarking Principles and Practices, pages 461–482. Marcel Dekker, 1999.

[13] S. A. Craver, M. Wu, B. Liu, A. Stubblefield, B. Swartzlander, and D. S. Wallach. Reading between the lines: Lessons from the SDMI challenge. In *10th USENIX Security Symposium*, Washington DC, 2001.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 99(1):1–38, 1977.

[15] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[16] H. Farid. Detecting digital forgeries using bispectral analysis. Technical Report AIM-1657, Massachusetts Institue of Technology, August 1999.

[17] H. Farid. Creating and detecting doctored and virtual images: Implications to the child pornography prevention act. Technical Report TR2004-518, Dartmouth College, September 2004.

[18] H. Farid and S. Lyu. Higher-order wavelet statistics and their application to digital forensics. In *IEEE Workshop on Statistical Analysis in Computer Vision*, Madison, Wisconsin, 2003.

[19] R. Fisher. The use of multiple measures in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[20] W. T. Freeman. Median filter for reconstructing missing color samples. US Patent, 4724395, 1988.

[21] J. Fridrich. Security of fragile authentication watermarks with localization. In *Proceedings of SPIE, Electronic Imaging 2002, Security and Watermarking of Multimedia Contents*, volume 4675, pages 691–700, January 2002.

[22] J. Fridrich and M. Goljan. Images with self-correcting capabilities. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 792–796, 1999.

[23] J. Fridrich, M. Goljan, and M. Du. Invertible authentication. In *Proceedings of SPIE, Security and Watermarking of Multimedia Contents*, 2001.

[24] J. Fridrich, D. Soukal, and J. Lukáš. Detection of copy-move forgery in digital images. In *Proceedings of Digital Forensic Research Workshop*, August 2003.

[25] G. L. Friedman. The trustworthy camera: Restoring credibility to the photographic image. *IEEE Transactions on Consumer Electronics*, 39(4):905–910, 1993.

[26] R. C. Gonzalez and R. E. Wood. *Digital Image Processing*. Addison Wesley Publishing Company, 1992.

[27] B. K. Gunturk, Y. Altunbasak, and R. M. Mersereau. Color plane interpolation using alternating projections. *IEEE Transactions on Image Processing*, 11(9):997–1013, September 2002.

[28] J. F. Hamilton and J. E. Adams. Adaptive color plan interpolation in single sensor color electronic camera. US Patent, 5629734, 1997.

[29] K. Hirakawa and T. W. Parks. Adaptive homogeneity-directed demosaicing algorithm. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 669–672, September 2003.

[30] C. W. Honsinger, P. Jones, M. Rabbani, and J. C. Stoffel. Lossless recovery of an original image containing embedded data. U.S. Patent Application, Docket No. 77102/E-D, 1999.

[31] S. Katzenbeisser and F. A. P. Petitcolas. *Information Techniques for Steganography and Digital Watermarking*. Artec House, 2000.

[32] R. G. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-29(6):1153–1160, December 1981.

[33] D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE Signal Processing Magazine*, 13(3):43–64, 1996.

[34] D. Kundur and D. Hatzinakos. Semi-blind image restoration based on tell-tale watermarking. In *Conference Record of the 32nd Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 933–937, 1998.

[35] D. Kundur and D. Hatzinakos. Digital watermarking for tell-tale tamper proofing and authentication. *Proceedings of the IEEE*, 87(7):1167–1180, 1999.

[36] C. A. Laroche and M. A. Prescott. Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients. US Patent, 5373322, 1994.

[37] C.-Y. Lin and S.-F. Chang. A robust image authentication algorithm surviving JPEG lossy compression. In *SPIE Storage and Retrieval of Image/Video Databases*, volume 3312, pages 296–307, 1998.

[38] E. T. Lin, C. I. Podilchuk, and E. J. Delp. Detection of image alterations using semi-fragile watermarks. In *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents II*, volume 3971, pages 152–163, 2000.

[39] J. Lukáš and J. Fridrich. Estimation of primary quantization matrix in double compressed JPEG images. In *Digital Forensic Research Workshop*, Cleveland, Ohio, August 2003.

[40] S. Lyu and H. Farid. How realistic is photorealistic? *IEEE Transactions on Signal Processing*, 2005. (in press).

[41] B. M. Macq and J.-J. Quisquater. Cryptology for digital TV broadcasting. *Proceedings of the IEEE*, 83(6):944–957, 1995.

[42] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[43] R. Matzner. An SNR estimation algorithm for complex baseband signals using higher order statistics. *Facta Universitatis (Nis)*, 6(1):41–52, 1993.

[44] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli. A DCT domain visible watermarking technique for images. In *IEEE International Conference on Multimedia and Expo*, volume 2, pages 1029–1032, 2000.

[45] P. Moulin and J. Liu. Analysis of multiresolution image denoising schemes using generalized gaussian and complexity priors. *IEEE Transactions on Information Theory*, 45(3):909–919, 1999.

[46] D. D. Muresan and T. W. Parks. Adaptively quadratic (AQua) image interpolation. *IEEE Transactions on Image Processing*, 13(5):690–698, May 2004.

[47] T.-T. Ng and S.-F. Chang. Blind detection of digital photomontage using higher order statistics. Technical Report ADVENT 201-2004-1, Columbia University, June 2004.

[48] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 1989.

[49] D. R. Pauluzzi and N. C. Beaulieu. A comparison of SNR estimation techniques for the awgn channel. *IEEE Transactions on Communications*, 48(10):1681–1691, 2000.

[50] A. C. Popescu and H. Farid. Exposing digital forgeries by detecting duplicated image regions. Technical Report TR2004-515, Dartmouth College, August 2004.

[51] A. C. Popescu and H. Farid. Statistical tools for digital forensics. In *Proceedings of the* 6^{th} *Information Hiding Workshop*, May 2004.

[52] A. C. Popescu and H. Farid. Exposing digital forgeries by detecting traces of re-sampling. *IEEE Transactions on Signal Processing*, 53(2), February 2005.

[53] A. C. Popescu and H. Farid. Exposing digital forgeries in color filter array interpolated images. *IEEE Transactions on Signal Processing*, 2005. (in review).

[54] JJ2000 partners. JJ2000, a Java™ implementation of JPEG2000. `http://jj2000.epfl.ch/jj_whitepaper/index.html`.

[55] R. Ramanath, W. E. Snyder, G. L. Bilbro, and W. A. Sander III. Demosaicking methods for Bayer color arrays. *Journal of Electronic Imaging*, 11(3):306–315, July 2002.

[56] M. Schneider and S.-F. Chang. A robust content-based digital signature for image authentication. In *IEEE International Conference on Image Processing*, volume 2, pages 227–230, 1996.

[57] E. P. Simoncelli and E. H. Adelson. Noise removal via bayesian wavelet coring. In *Proceedings of the 3rd International Conference on Image Processing*, pages 379–382, Lausanne, Switzerland, September 1996.

[58] D. Storck. A new approach to integrity of digital images. In *IFIP Conference on Mobile Communication*, pages 309–316, 1996.

[59] G. K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 1991.

[60] P. W. Wong. A public key watermark for image verification and authentication. In *IEEE International Conference on Image Processing*, volume 1, pages 455–459, 1998.

[61] P. W. Wong. A watermark for image integrity and ownership verification. In *Proceedings of IS&T PIC Conference*, pages 374–379, Portland, Oregon, May 1998.

[62] C. F. J. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[63] M. Wu and B. Liu. Watermarking for image authentication. In *IEEE International Conference on Image Processing*, volume 2, pages 437–441, 1998.

[64] M. Yeung and F. Mintzer. An invisible watermarking technique for image verification. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 680–683, 1997.

[65] G.-J. Yu, C.-S. Lu, H.-Y. M. Liao, and J.-P. Sheu. Mean quantization blind watermarking for image authentication. In *IEEE International Conference on Image Processing*, volume 3, pages 706–709, 2000.