ABSTRACT

Title of dissertation:	DIGITAL MULTIMEDIA FORENSICS AND ANTI-FORENSICS
	Matthew C. Stamm, Doctor of Philosophy, 2012
Dissertation directed by:	Professor K. J. Ray Liu Department of Electrical and Computer Engineering

As the use of digital multimedia content such as images and video has increased, so has the means and the incentive to create digital forgeries. Presently, powerful editing software allows forgers to create perceptually convincing digital forgeries. Accordingly, there is a great need for techniques capable of authenticating digital multimedia content. In response to this, researchers have begun developing digital forensic techniques capable of identifying digital forgeries. These forensic techniques operate by detecting imperceptible traces left by editing operations in digital multimedia content. In this dissertation, we propose several new digital forensic techniques to detect evidence of editing in digital multimedia content.

We begin by identifying the fingerprints left by pixel value mappings and show how these can be used to detect the use of contrast enhancement in images. We use these fingerprints to perform a number of additional forensic tasks such as identifying cut-and-paste forgeries, detecting the addition of noise to previously JPEG compressed images, and estimating the contrast enhancement mapping used to alter an image.

Additionally, we consider the problem of multimedia security from the forger's point of view. We demonstrate that an intelligent forger can design *anti-forensic* operations to hide editing fingerprints and fool forensic techniques. We propose an anti-forensic technique to remove compression fingerprints from digital images and show that this technique can be used to fool several state-of-the-art forensic algorithms. We examine the problem of detecting frame deletion in digital video and develop both a technique to detect frame deletion and an anti-forensic technique to hide frame deletion fingerprints. We show that this anti-forensic operation leaves behind fingerprints of its own and propose a technique to detect the use of frame deletion anti-forensics. The ability of a forensic investigator to detect both editing and the use of anti-forensics results in a dynamic interplay between the forger and forensic investigator. We use develop a game theoretic framework to analyze this interplay and identify the set of actions that each party will rationally choose. Additionally, we show that anti-forensics can be used protect against reverse engineering. To demonstrate this, we propose an anti-forensic module that can be integrated into digital cameras to protect color interpolation methods.

DIGITAL MULTIMEDIA FORENSICS AND ANTI-FORENSICS

by

Matthew C. Stamm

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Professor K. J. Ray Liu, Chair/Advisor
Professor Min Wu
Professor Rama Chellappa
Professor Dianne P. O'Leary
Professor Lawrence C. Washington

© Copyright by
Matthew C. Stamm
2012

Dedication

This dissertation is dedicated to my parents. They instilled in me a sense of curiosity that has helped me choose my path in life. Without their love, support, and encouragement, I would never have had the ability to walk it.

Acknowledgments

I wish to begin by expressing my deepest thanks to my advisor, Prof. K. J. Ray Liu. His encouragement helped me decide to pursue a Ph. D. and he has been a constant source of guidance and support throughout my graduate career. He has pushed me to achieve more than I thought possible and thanks to him I have grown tremendously both intellectually and professionally. Most importantly, he has set an example of the researcher, educator, and mentor that I aspire to be.

I would like to thank the members of my dissertation committee; Prof. Min Wu, Prof. Rama Chellappa, Prof. Dianne O'Leary, and Prof. Lawrence Washington. I would like specially thank Prof. Wu and Prof Chellappa for acting as references for me. I would also like to thank Prof. Adrian Papamarcou, Prof. Jonathan Simon, and Prof. Steven Marcus for acting as references, helping with various applications, and assisting me throughout my graduate career.

I would like to thank the members of the SIG and MAST groups, especially Dr. Wan-Yi Lin, Dr. Steven Tjoa, Dr. Yan Chen, Xiaoyu Chu, and Zhung-Han Wu, along with Dr. Ashwin Swaminathan, Dr. Avinash Varna, and Wei-Hong Chuang for their friendship and interesting research discussions.

I would also like to thank all of my friends and family for believing in me, encouraging me, and brightening my life. Special thanks are due to Lauren White for her love and support. She lifted my spirits when I was down, constantly encouraged me, and has made the past year one of the best of my life.

Most of all, I would like to thank my parents. Their unwaivering support and encouragement has helped me follow my dreams, and my love of learning is due directly to them. For every trip to the science center, every nature walk they took me on, every science experiment they performed with me, and every household appliance which I took apart but could not put back together, I thank them. This dissertation is as much theirs as it is mine.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In recent years, digital multimedia content has become ubiquitous throughout society. High quality audio recorders along with digital image and video cameras allow anyone to capture digital multimedia signals. Landline and mobile Internet access allows users to access digital content virtually anywhere and at any time. These conditions have led many governmental, legal, scientific, and news media organizations to rely on digital multimedia content to make critical decisions, convey information, and use as evidence of specific events.

Widespread reliance on digital content proves to be problematic, however, as the rise of digital media has coincided with the widespread availability of digital editing software. At present, a forger can easily manipulate digital content such as images or video to create perceptually realistic forgeries. Take, for example, the visually convincing image forgery shown in Fig. 1.1. This falsified image, which was released by the Iranian government in 2008, was edited to cover up an unsuccessful missile launch [31]. While this forgery was identified through careful visual inspection, more sophisticated forgeries are likely to remain unnoticed. As a result, many organizations desire some means of authenticating digital multimedia content to avoid both embarrassment and legal ramifications.

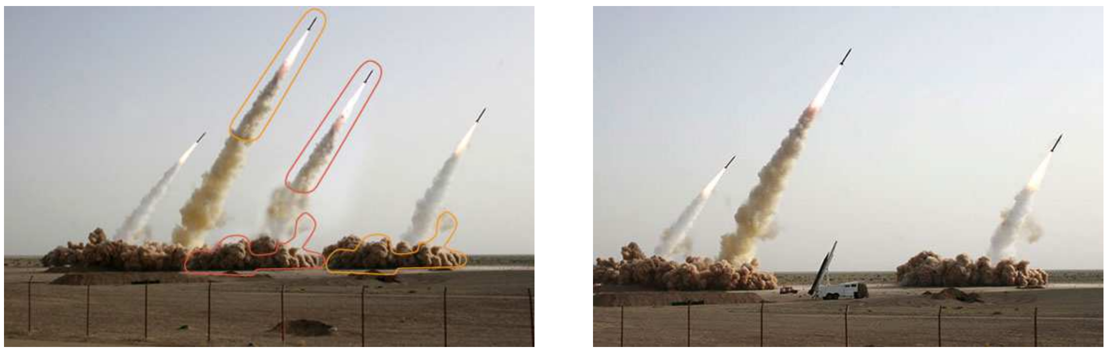


Figure 1.1: Left: A digital image forgery of a missile test launch released by the Iranian government. The falsified image regions are outlined. Right: The unaltered image used to create the forgery [31].

Previously, digital watermarking techniques have been proposed as a means of providing multimedia security [10]. For watermarking techniques to be successful, however, an extrinsic watermark must be inserted into the digital content by a trusted source before any manipulation occurs. This is unrealistic in many scenarios, because the party that captures the digital content can alter it before inserting the watermark. Similarly, encryption techniques can prevent an unauthorized user from accessing and altering a digital multimedia file, but they cannot prevent a file's creator from manipulating it before encryption.

In response to these challenges, the field of *digital multimedia forensics* has been born. Digital multimedia forensics involves the study and development of techniques to determine the authenticity, processing history, and origin of digital multimedia content without relying on any information aside from the digital content itself. This is done by making use of the fact that most signal processing operations leave behind perceptually undetectable traces known as *fingerprints* in digital content similar to the way that a criminal leaves behind fingerprints at a crime scene.

By discovering these fingerprints and developing techniques to detect them, digital forensics researchers can identify digital multimedia forgeries. Because most signal processing operations leave behind unique fingerprints, no universal method of detecting digital forgeries exists. Instead, many forensic tests must be designed to identify the fingerprints of a wide variety of digital content editing operations. It has been posited that if a large set of forensic methods is developed, it will be difficult for a forger to create a digital forgery capable of fooling all forensic authentication techniques [5].

Though existing digital forensic techniques are capable of detecting several standard digital media manipulations, they do not account for the possibility that *anti-forensic* operations designed to hide traces of manipulation may be applied to digital content. In reality, it is quite possible that a forger with a digital signal processing background may be able to secretly develop anti-forensic operations and use them to create undetectable digital forgeries. As a result, several multimedia forensic techniques may possess vulnerabilities that are unknown to the forensic community at large.

To protect against this scenario, it is crucial for researchers to develop and study anti-forensic operations so that vulnerabilities in existing forensic techniques may be known. This will help researchers to know when digital forensic results can be trusted and may assist researchers in the development of improved digital forensic techniques. Furthermore, the study of anti-forensic operations can also lead to the identification of fingerprints left by anti-forensic operations and the development of techniques capable of detecting when an anti-forensic operation has been used to hide evidence forgery.

It is clear that the authentication of multimedia signals poses a great challenge to information security researchers. Not only must new forensic techniques be developed, but anti-forensic techniques must also be uncovered and their effects mitigated. The reactions of forgers to the development of more sophisticated forensic methods must be predicted and the dynamic interplay between a forger and forensic investigator must be understood. Additionally, unintended uses of forensic techniques must be anticipated and protected against. In this dissertation, we address these problems and show how information security can be provided through the study of both digital forensics and anti-forensics.

1.2 Dissertation Outline and Contributions

From the above discussion, we can clearly see the need for forensic techniques capable of authenticating digital multimedia signals. In this dissertation, we propose several new forensic techniques designed to detect the use of a variety of multimedia editing operations. Furthermore, we take the novel step of considering multimedia signal authentication from the point-of-view of the forger. We propose a set of anti-forensic operations and demonstrate that a forger can use them to fool state-of-the-art forensic techniques. We show how both a forger and forensic investigator can respond to the actions of each other, and develop a game theoretic framework to understand the dynamic interaction between these two parties. Additionally, we show how anti-forensics can be used to prevent the reverse engineering of digital devices with forensic techniques. The rest of this dissertation is organized as follows.

1.2.1 Image Forgery Detection Using Statistical Intrinsic Fingerprints (Chapter 2)

In order to compensate for poor or undesirable lighting conditions, a forger will often perform contrast enhancement on an image. Contrast enhancement encompasses a number of widely used image editing operations such as gamma correction or histogram equalization. Each of these techniques operate by applying a nonlinear mapping to an image's pixel values.

In this chapter, we show that pixel value mappings leave behind statistical traces, which we shall refer to as a mapping's *intrinsic fingerprint*, in an image's pixel value histogram. We develop a model of the forensically significant properties of the histogram of an unaltered image and use this model to identify diagnostic features of a mapping's intrinsic fingerprint. We then propose forensic techniques for detecting general forms of globally and locally applied contrast enhancement. We demonstrate that localized contrast enhancement can be used to identify cut-and-paste forgeries. We identify the specific intrinsic fingerprints of histogram equalization and propose a method to detect if histogram equalization was used to perform contrast enhancement on an image.

Additionally, we propose a method to detect the global addition of noise to a previously JPEG compressed image. We do this by observing that the intrinsic fingerprint of a specially chosen pixel value mapping will be altered if it is applied to an image's pixel values after the addition of noise. Through a number of simulations, we test the efficacy of each proposed forensic technique. Our simulation results show that aside from exceptional cases, all of our detection methods are able to correctly detect the use of their designated image processing operation with a probability of 99% given a false alarm probability of 7% or less.

1.2.2 Forensic Estimation of Contrast Enhancement Mappings (Chapter 3)

Once evidence of editing has been identified in a digital multimedia file, a series of new questions arise for a forensic investigator. For example, a forensic investigator may wish to learn specific details of how the file was altered and what parameters the forger used when applying the editing operation. Additionally, the forensic investigator may wish to ascertain as much information as possible about the multimedia file before it was altered.

In this chapter, we address these questions in context of contrast enhanced digital images. We propose an iterative algorithm to jointly estimate any arbitrary contrast enhancement mapping used to modify an image as well as the pixel value histogram of the image before contrast enhancement. To do this, we use a probabilistic model of an image's pixel value histogram to determine which histogram entries are most likely to correspond to contrast enhancement artifacts. Experimental results are presented to demonstrate the effectiveness of our proposed method.

1.2.3 Digital Image Compression Anti-Forensics (Chapter 4)

As society has become increasingly reliant upon digital images to communicate visual information, a number of forensic techniques have been developed to verify the authenticity of digital images. Amongst the most successful of these are techniques that make use of an image's compression history and its associated compression fingerprints. Little consideration has been given, however, to *anti-forensic* techniques capable of fooling forensic algorithms. In this chapter, we present a set of anti-forensic techniques designed to remove forensically significant indicators of compression from an image. We do this by first developing a generalized framework for the design of anti-forensic techniques to remove compression fingerprints from an image's transform coefficients. This framework operates by estimating the distribution of an image's transform coefficients before compression, then adding *anti-forensic dither* to the transform coefficients of a compressed image so that their distribution matches the estimated one. We then use this framework to develop anti-forensic techniques specifically targeted at erasing compression fingerprints left by both JPEG and wavelet-based coders.

Additionally, we propose a technique to remove statistical traces of the blocking artifacts left by image compression algorithms that divide an image into segments during processing. Through a series of experiments, we demonstrate that our anti-forensic techniques are capable of removing forensically detectable traces of image compression without significantly impacting an image's visual quality. Furthermore, we show how these techniques can be used to render several forms of image tampering such as double JPEG compression, cut-and-paste image forgery, and image origin falsification undetectable through compression history based forensic means.

1.2.4 Temporal Forensics and Anti-Forensics for Digital Video (Chapter 5)

Due to the ease with which digital information can be altered, many digital forensic techniques have been developed to authenticate multimedia content. Similarly, a number of anti-forensic operations have recently been designed to make digital forgeries undetectable by forensic techniques. However, like the digital manipulations they are designed to hide, many anti-forensic operations leave behind their own forensically detectable traces. As a result, a digital forger must balance the tradeoff between completely erasing evidence of their forgery and introducing new evidence of anti-forensic manipulation. Because a forensic investigator is typically bound by a constraint on the probability of false alarm (P_{fa}), the accuracy of detecting forgeries must be balanced with the accuracy of detecting the use of anti-forensics.

In this chapter, we analyze the interaction between a forger and a forensic investigator by examining the problem of authenticating digital videos. Specifically, we study the problem of adding or deleting a sequence of frames from a digital video. We begin by developing a theoretical model of the forensically detectable fingerprints that frame deletion or addition leaves behind, then use this model to improve upon the video frame deletion or addition detection technique proposed by

Wang and Farid. Next, we propose an anti-forensic technique designed to fool video forensic techniques and develop a method for detecting the use of anti-forensics. We introduce a new set of techniques for evaluating the performance of anti-forensic operations and develop a game theoretic framework for analyzing the interplay between a forensic investigator and a forger. We use these new techniques to evaluate the performance of each of our proposed forensic and anti-forensic techniques, and identify the optimal actions of both the forger and forensic investigator.

1.2.5 Protection Against Reverse Engineering in Digital Cameras Using Anti-Forensics (Chapter 6)

One important set of forensic techniques operates by estimating signal processing components of a digital camera's signal processing pipeline, then using these estimates to perform forensic tasks such as camera identification or forgery detection. However, because these techniques are capable of estimating a camera's internal signal processing components, these forensic techniques can be used for reverse engineering. In this chapter, we propose integrating an anti-forensic module into a digital camera's processing pipeline to protect against forensic reverse engineering. Our proposed technique operates by removing linear dependencies amongst an output image's interpolated color values and by disrupting the color sampling grid. Experimental results show that our proposed technique can be effectively used to protect against the forensic reverse engineering of key components of a digital camera's processing pipeline.

Chapter 2

Forensic Detection of Image Manipulation Using Statistical Intrinsic Fingerprints

One of the primary goals of digital image forensics is the identification of images and image regions which have undergone some form of manipulation or alteration. Because of the ill-posed nature of this problem, no universal method of detecting image forgeries exists. Instead, a number of techniques have been proposed to identify image alterations under a variety of scenarios. While each of these methods possesses their own limitations, it has been posited that if a large set of forensic methods are developed, it will be difficult for a forger to create an image capable of fooling all image authentication techniques [5].

Previous image forensic work has dealt with the identification of computer generated objects within an image [29] as well as detecting lighting angle inconsistencies [20], [18]. Inconsistencies in chromatic aberration [19] as well as the absence of CFA interpolation induced correlations [35] have been used to identify inauthentic regions of an image. Classifier based approaches have been proposed which identify image forgeries using a variety of statistical features [30], [2], [1]. Though these techniques are capable of detecting that an image has undergone some form of manipulation, they are unable to determine how an image has been altered beyond the identification of manipulated image regions.

One set of digital forensic techniques aimed at detecting image tampering has grown out of research into imaging device identification. Forensic imaging device identification methods attempt to determine the type of device used to capture an image, ascertain the device manufacturer or model, and identify the particular imaging device used [53]. These methods generally perform identification by estimating some device specific parameter such as color filter array (CFA) interpolation coefficients or sensor noise. Image forgery detection techniques have been proposed which operate by locating inconsistencies in these parameters [5], [27], or by using these parameters to estimate a tampering filter [52]. While these techniques are quite effective, they too suffer the drawback of being unable to identify the use of specific image altering operations.

It is important to note that most image altering operations leave behind distinct, traceable "fingerprints" in the form of image alteration artifacts. Because these fingerprints are often unique to each operation, an individual test to catch each type of image manipulation must be designed. While detecting image forgeries using these techniques requires performing a large set of operation-specific tests, these methods are able to provide insight into the specific operations used to manipulate an image. Prior work which identifies image tampering by detecting operation specific fingerprints includes the detection of resampling [34], double JPEG compression [33], [32], [28], as well as the parameterization of gamma correction [8]. Methods for detecting image forgeries have been proposed by detecting local abnormalities in an image's signal to noise ratio [33]. Additionally, the efficient identification of copy and move forgeries has been studied [11].

In this work, we show that with the exception of the identity mapping, pixel value mappings leave behind statistical artifacts which are visible in an image's pixel value histogram. We refer to these artifacts as the *intrinsic fingerprint* of a pixel value mapping. By observing the common properties of the histograms of unaltered images, we are able to build a model of an unaltered image's pixel value histogram. We then use this model to identify diagnostic features of a pixel value mapping's intrinsic fingerprint. Because a number of image processing operations are in essence pixel value mappings, we propose a set of image forgery detection techniques which operate by detecting the intrinsic fingerprint of each operation. Specifically, we propose methods for detecting general forms of globally and locally applied contrast enhancement, as well as a method for identifying the use of histogram equalization, a commonly used form of contrast enhancement. Additionally, we propose a method to detect the global addition of noise to a previously JPEG compressed image by detailing the effect of noise on the fingerprint of a known pixel value mapping applied to the image in question.

While much of this work focuses on detecting operations which alter the perceptual qualities of an image as opposed to more obviously malicious tampering, detecting the image manipulations discussed in this work is still forensically significant. The detection of globally applied contrast enhancement provides insight into an image's processing history and may be useful prior information for other detection algorithms. Furthermore, contrast enhancement operations may be locally applied to disguise visual clues of image tampering. Localized detection of these operations can be used as evidence of cut-and-paste type forgery. Additive noise may be globally applied to an image not only to cover visual evidence of forgery, but also in an attempt to destroy forensically significant indicators of other tampering operations. Though the detection of these types of operations may not necessarily pertain to malicious tampering, they certainly throw in doubt the authenticity of the image and its content.

This chapter is organized as follows. In Section 2.1, we describe the forensically significant qualities of an unaltered image's pixel value histogram. In Section 2.2 we define the intrinsic fingerprint of a pixel value mapping. We describe our proposed contrast enhancement detection techniques in Section 2.3. Included are methods for detecting both globally and locally applied contrast enhancement as well as a method for identifying histogram equalization. We develop a method for detecting the addition of noise to a previously JPEG compressed image in Section 2.4. Experiments designed to test the efficacy of each forensic scheme as well as simulation results are discussed after each detection method is proposed. We summarize this chapter in Section 2.5.

2.1 System Model and Assumptions

In this work, we consider digital images created by using an electronic imaging device to capture a real world scene. We adopt the following model of the digital capture process. Each pixel is assigned a value by measuring the light intensity reflected from a real world scene onto an electronic sensor over the area pertaining to that pixel. Inherent in this process is the addition of some zero mean sensor noise which arises due to several phenomena including shot noise, dark current, and on-chip amplifier noise [17]. For color images, it is often the case that the light passes through a color filter array so that only one color component is measured at each pixel location in this fashion. If this is the case, the color components not observed at each pixel are determined through interpolation. At the end of this process, the pixel values are quantized, then stored as the unaltered image.

When analyzing a digital image, a histogram $h(l)$ of the color or gray level values l recorded at each pixel can be generated by creating L equally spaced bins which span the range of possible pixel values, then tabulating the number of pixels whose value falls within the range of each bin. Unless otherwise specified, we will hereafter assume that all gray level values lie in the set $\mathcal{P} = \{0, \ldots, 255\}$, all color values lie in the set $\mathcal{P}^3$, and that all pixel value histograms are calculated using 256 bins so that each bin corresponds to a unique gray or color layer value. After viewing the pixel value histograms of several camera generated images corresponding to a variety of scenes, we have observed that these histograms share common properties. None of the histograms contain sudden zeros or impulsive peaks. Furthermore, individual histogram values do not differ greatly from the histogram's envelope. To unify these properties, which arise due to observational noise [17], sampling effects, and complex lighting environments, we describe pixel value histograms as *interpolatably connected*. We denote an interpolatably connected histogram as one where any histogram value $h(l)$ can be approximated by $\hat{h}(l)$, the interpolated value of the histogram at pixel value l calculated using a cubic spline given $h(t)$ for all $t \in \mathcal{P} \setminus l$. The histogram of a typical unaltered image as well as its approximation

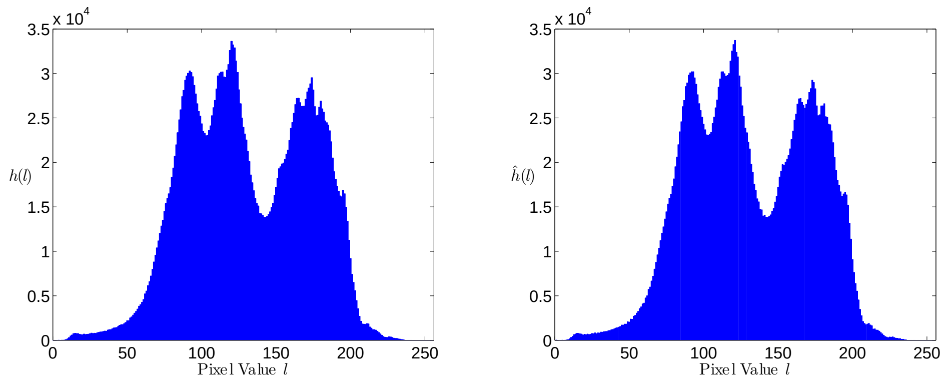


Figure 2.1: Left: Histogram of a typical image. Right: Approximation of the histogram at left by sequentially removing then interpolating the value of each histogram entry.

$\hat{h}$, where each value of $\hat{h}$ has been calculated by removing a particular value from h then interpolating this value using a cubic spline, are shown in Fig. 2.1. As can be seen in this example, there is very little difference between the image's histogram and its approximation.

To justify this model, we compiled a database of 341 unaltered images captured using a variety of digital cameras. We obtained each image's pixel value histogram h, as well as its approximated histogram $\hat{h}$, where each value $\hat{h}(x)$ was interpolated using cubic spline interpolation. We then calculated the mean squared error between $\hat{h}$ and h along with the signal power of h to obtain a signal to noise ratio (SNR). The mean SNR of all image's histograms in the test database was 30.67 dB, reinforcing the notion that an image's pixel value histogram can be modeled as an interpolatably connected function.

There does exist one naturally occurring phenomena, which we refer to as *histogram saturation*, that may cause an unaltered image's pixel value histogram to contain an impulsive peak at one of two possible locations. High end histogram

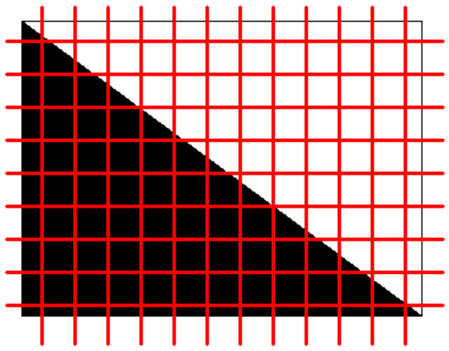


Figure 2.2: Image sampling effects example.

saturation effects occur in images corresponding to especially bright scenes where the dynamic range of the observed light intensity values extends well above the cutoff for the maximum pixel value. Because these pixels must be assigned the maximum pixel value of 255, a disproportionate number of pixels will take this value resulting in an impulsive peak at in the high end of an image's histogram. Low end saturation effects occur in unusually dark images, where a large number of pixels taking the value 0 will cause an impulsive peak to occur at the low end of an image's histogram. While low end histogram saturation occurs less frequently than high end saturation, we have observed it in several unaltered images.

To explain why our histogram model is appropriate for digital images, consider the simple case of imaging a scene consisting of two distinct color regions shown in Fig. 2.2. Instinctively, we might assume that the histogram of this image would consist of zeros everywhere except for two impulses located at the pixel values corresponding to each of the two colors present in this scene. Such a histogram would obviously violate our model. In this scenario, however, the border between the color regions does not align with the pixel boundaries on the sensor of the imaging device, denoted by the grid. Many pixels lying along the color border correspond to sensor areas containing both colors. The resulting values of each of these pixels will lie in the convex hull of the values corresponding to each of the two colors present in the scene. The introduction of these new pixel values will effectively 'smooth out' the pixel value histogram. Additionally, in the case of a color image, color values not observed at a particular pixel location must be interpolated because of the use of a CFA. The value of these interpolated pixels will also lie in the convex hull of their neighbors values and further smooth the histogram, resulting in one which is interpolatably connected.

Due to the complexity of real world scenes, it is exceedingly unlikely that the all color borders in an image will align directly with the pixel borders on an imaging device's sensor. Because of this, the effect described above should be present in virtually all real world images. Furthermore, additional factors contribute to the 'connectivity' of pixel value histograms of images captured by digital cameras. The complex nature of most natural and man-made lighting environments rarely result in a real world scene consisting of several distinct colors with no shading. Instead, a continuum of colors and illumination levels normally exist. Furthermore, the presence of observational noise will slightly change the value of several pixels during the image capture process, thus further smoothing the histogram and resulting in one which is interpolatably connected.

2.2 Statistical Intrinsic Fingerprints of Pixel Value Mappings

A number of image processing operations, such as contrast enhancement, either include or can be specified entirely by a pixel value mapping. As is the case with most image processing operations, pixel value mappings leave behind distinct, forensically significant artifacts. These artifacts, which we will refer to as the *intrinsic fingerprint* of a pixel value mapping m, manifest themselves primarily in an image's pixel value histogram. To understand the effect of a pixel value mapping on an image's histogram, let us define $x \in \mathcal{P}$ as a pixel value present in an unaltered image and $y \in \mathcal{P}$ as the value that m maps x to such that

$$y = m(x). \tag{2.1}$$

Using this equation, the relationship between the pixel value histogram h_X of the unaltered image and the pixel value histogram h_Y of the same image after its pixel values have been subjected to the mapping m can be written as

$$h_Y(l) = \sum_{t=0}^{255} h_X(t)\, \mathbb{1}(m(t) = l), \tag{2.2}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function. As a consequence, all entries in h_Y must take a value of either zero or the sum of several entries in h_X. Furthermore, any time n unaltered pixel values are mapped to the same output value, $n-1$ entries in h_Y must take a value of zero.

We now define the intrinsic fingerprint of m as

$$\begin{aligned} f_m(l) &= h_Y(l) - h_X(l) \\ &= \sum_{t=0}^{255} h_X(t)\, \mathbb{1}(m(t) = l) - h_X(l), \end{aligned} \tag{2.3}$$

which represents the change in the image's pixel value histogram due to the application of the mapping m. We can see that though the pixel value mapping is deterministic, its fingerprint depends on the image's histogram statistics. In subsequent sections it will be useful to examine a frequency domain representation of $f_m(l)$. Letting $\hat{m}(l) = m(l) - l$, the discrete Fourier transform (DFT) of $f_m(l)$ can be written as

$$\begin{aligned} F_m(k) &= \mathrm{DFT}\{f_m(l)\} \\ &= \sum_{l=0}^{255} h_X(l) \left(e^{-j\frac{\pi k \hat{m}(l)}{128}} - 1 \right) e^{-j\frac{\pi k l}{128}} \\ &= -2j \sum_{l=0}^{255} h_X(l) \sin\left(k \frac{\pi \hat{m}(l)}{256} \right) e^{-j\frac{\pi k}{128}\left(\frac{\hat{m}(l)}{2}+l\right)}. \end{aligned} \tag{2.4}$$

By examining equations (2.3) and (2.4), we can see that the intrinsic fingerprint is characterized not only by $m(l)$, but by $h_X(l)$ as well. Despite this, the intrinsic fingerprints left in two images with different pixel value histograms will be quite similar. In the frequency domain, a mapping's tampering fingerprint consists of a linear combination of sinusoids whose frequencies are determined by $\hat{m}(l)$, which is nonzero only when $m(l) \neq l$. While the value of $h_X(l)$ affects the weight of each sinusoid in the summation, the presence and frequency of each sinusoid, and hence the basic structure of the intrinsic fingerprint, is determined by m.

Fig. 2.3 shows an example illustrating the similarity between fingerprints left in images with different pixel value histograms by a common mapping. As a reference, the pixel value histograms of a synthesized image with a uniform pixel values distribution and a typical image captured by a digital camera are shown in Figs. 2.3(a) and (b) respectively. The DFT of both histograms are shown in Figs.

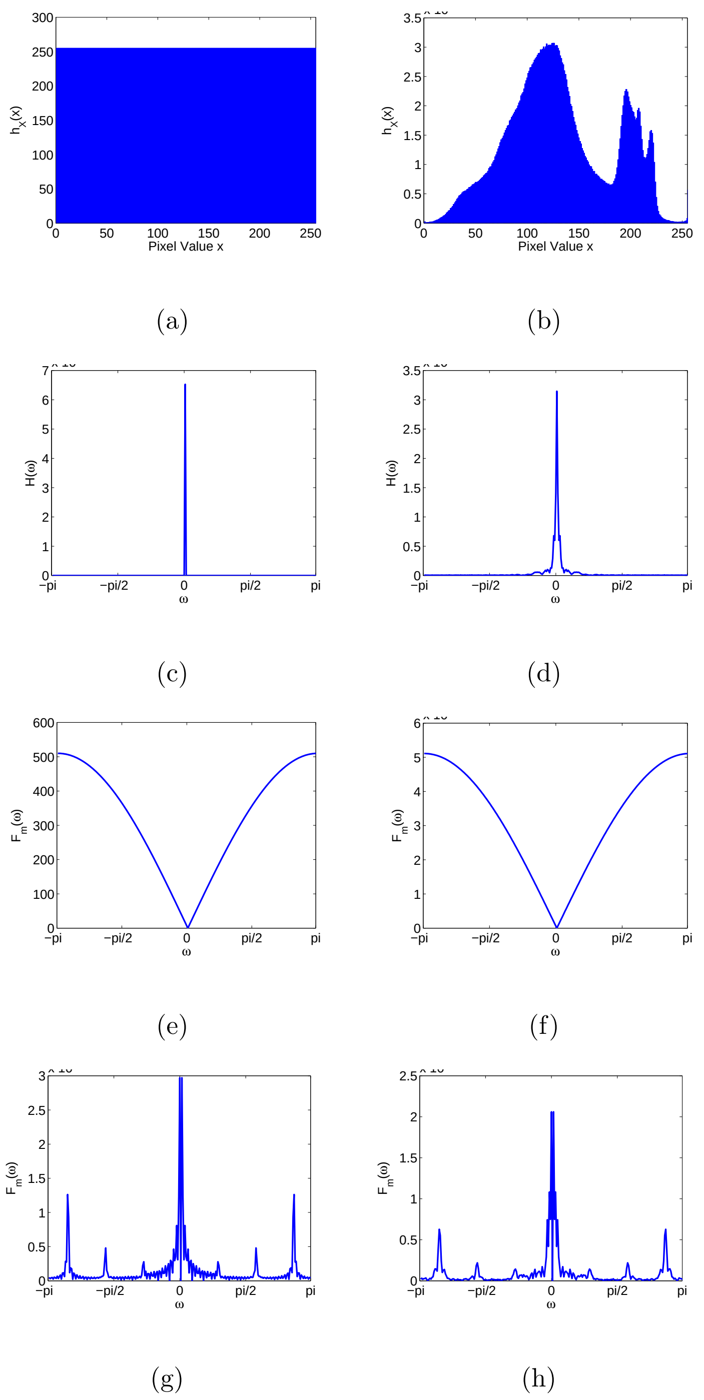


Figure 2.3: Tampering fingerprint example showing the pixel value histograms of (a) a synthesized image with a uniform pixel distribution and (b) a real world image captured with a digital camera, the magnitude of the DFT of the histogram of (c) the synthesized image and (d) the real image, the magnitude of the frequency domain tampering fingerprints of (2.5) left in (e) the synthesized image and (f) the real image, as well as the magnitude of the frequency domain intrinsic fingerprints of (2.6) left in (e) the synthesized image and (f) the real image.

2.3(c) and (d). As can be seen, these histograms differ significantly in both the pixel value and frequency domains. Next, the intrinsic fingerprints left in each image's histogram by the mapping

$$m(l) = \begin{cases} l & \text{if } l \neq 100 \\ l+1 & \text{if } l = 100, \end{cases} \tag{2.5}$$

are compared. This mapping alters only one pixel value and is one of the simplest possible pixel value mappings. In this case, the intrinsic fingerprint left in each image's histogram will differ only by a scaling factor. This can be seen in Figs. 2.3(e) and (f), which show the magnitude of the frequency domain representation of each fingerprint. Finally, the intrinsic fingerprints left by the pixel value mapping

$$m(l) = \text{round}(\tfrac{7}{11}l), \tag{2.6}$$

are compared. This mapping is more complex than the previously considered mapping, and affects several pixel values. Figs. 2.3(g) and (h) show the magnitude of the frequency domain representation of each fingerprint. Though these fingerprints differ by more than a simple scaling factor, they share several identifying features including local peaks at $\omega = \pm 0.9081$, ± 1.8162, and ± 2.7243 radians, where

$$\omega = \begin{cases} \frac{k\pi}{128} & \text{if } 0 \leq k < 128, \\ \frac{(k-256)\pi}{128} & \text{if } 128 \leq k \leq 255. \end{cases} \tag{2.7}$$

In subsequent sections, we use intrinsic fingerprints along with our histogram model to identify evidence of image tampering. When examining a potentially altered image, if the histogram of unaltered pixel values is known, the tampering fingerprint can be obtained using (2.3). If the tampering fingerprint is zero for all

l, one can conclude that a pixel value mapping was not used to alter the image. Alternatively, if the tampering fingerprint is nonzero for any values of l, it can be used to help determine the mapping used to alter the image. In most real scenarios, however, one has no *a priori* knowledge of an image's pixel value histogram, thus the tampering fingerprint cannot be calculated. Despite this, we are able to ascertain the presence of a tampering fingerprint by determining identifying features of a mapping's intrinsic fingerprint and searching for their presence in the histogram of the image in question. Furthermore, we reduce the number of false detections by using our histogram model to separate naturally occurring histogram features from those which correspond to a pixel value mapping's intrinsic fingerprint.

It is important to note that the concept of an intrinsic fingerprint extends to any monotonically increasing mapping, aside from the identity mapping, applied to discrete-valued data. For example, when an image undergoes double JPEG compression, its DCT coefficients are doubly quantized according to the mapping $y = q_2 \, \mathrm{round}(\frac{q_1}{q_2} \, \mathrm{round}(\frac{x}{q_1}))$ where q_1 and q_2 are the quantization steps used. The periodic DCT coefficient histogram artifacts used in [33], [32], and [28] to identify double JPEG compression correspond to key features of the intrinsic fingerprint of this mapping. In fact, any time that an identifying feature of a mapping's intrinsic fingerprint can be determined, it can be used to detect the application of that mapping.

2.3 Detecting Contrast Enhancement

In this section, we identify the intrinsic fingerprints of contrast enhancement mappings and use them to develop a set of image forensic techniques capable of detecting if an image has undergone contrast enhancement. While prior image forensic work has studied gamma correction [8], [33], this work assumes that the forensic examiner knows which specific type of contrast enhancement may have been applied and that the contrast enhancement mapping can be described by a simple parametric equation. Here, we present a detection approach which can be used to detect more general contrast enhancement operations and which requires no *a priori* knowledge of the form of contrast enhancement potentially applied. We begin by discussing a method for detecting the global application of contrast enhancement which operates by identifying histogram features indicative of general contrast enhancement fingerprints [40]. Next, we extend this technique into one capable of detecting locally applied contrast enhancement and show how it can be used to detect certain cut-and-paste image forgeries. Additionally we present a method for identifying the use of histogram equalization, a specific form of contrast enhancement, by identifying histogram features unique to its intrinsic fingerprint.

2.3.1 Detection of Globally Applied Contrast Enhancement

Contrast enhancement operations seek to increase the dynamic range of pixel values within an image. Most globally applied contrast enhancement operations accomplish this by applying a nonlinear mapping to the values of each pixel in the image, as described in Section 2.2. In order to detect these operations, we must therefore detect the use of any pixel value mapping employed by a contrast enhancement operation. Without excluding any commonly used forms of contrast enhancement, we assume that all pixel value mappings in question are monotonically increasing. By considering only monotonic pixel value mappings, we purposefully exclude mappings which consist of a simple reordering of pixel values. As was previously mentioned, we detect the use of global contrast enhancement by identifying a characteristic feature of all monotonically increasing pixel value mappings (excluding the identity mapping), then use this feature in conjunction with our histogram model to ascertain whether the pixel value histogram of an image corresponds to a contrast enhanced image or an unaltered one.

In order to identify a diagnostic feature for contrast enhancement operations, let us first consider the effect of applying the mapping $m_{\tau+}$, defined as

$$m_{\tau+}(l) = \begin{cases} l & \text{if } l \neq \tau, \\ l+1 & \text{if } l = \tau, \end{cases} \tag{2.8}$$

to an image with pixel value histogram h_X, resulting in an altered image with pixel value histogram h_Y. This mapping is significant because any monotonically increasing pixel value mapping, aside from the identity mapping, can be formed by the proper composition of the mappings $m_{\tau+}$ and $m_{\tau-}$ using various values of τ, where $m_{\tau-}$ is defined as

$$m_{\tau-}(l) = \begin{cases} l & \text{if } l \neq \tau, \\ l-1 & \text{if } l = \tau. \end{cases} \tag{2.9}$$

Furthermore, let $h_X(\tau) = a$ and $h_X(\tau+1) = b$; therefore after the mapping $m_{\tau+}$ is applied to the image, the altered image's histogram values at τ and $\tau+1$ will be $h_Y(\tau)=0$ and $h_Y(\tau+1)=a+b$. The square of the Euclidean norm of h_X, denoted by $\|h_X\|_2^2$, will be less than that of h_Y because

$$\begin{aligned}\|h_X\|_2^2 &= \sum_l h_X(l)^2 \\ &= \sum_{l\neq\tau,\tau+1} h_X(l)^2 + a^2 + b^2 \\ &\leq \sum_{l\neq\tau,\tau+1} h_X(l)^2 + (a+b)^2 \\ &= \|h_Y\|_2^2.\end{aligned} \tag{2.10}$$

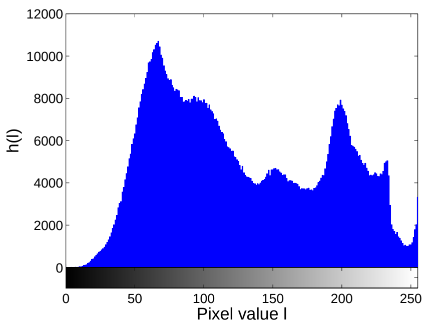
By Parseval's theorem, the energy of the DFT of h_Y must be greater than or equal to the energy of the DFT of h_X, however, this increase in energy cannot be realized in the DC coefficient because the total number of pixels in the image remains constant. An identical result can be proved for the mapping $m_{\tau-}$.

Because all monotonically increasing contrast enhancement mappings can be formed using the proper composition of the mappings $m_{\tau+}$ and $m_{\tau-}$, all contrast enhancement mappings result in an increase in energy within the image's pixel value histogram. This increase in energy corresponds to the energy of the intrinsic fingerprint left by the contrast enhancement mapping. In our experiments, we have observed that the increase in energy tends to be spread across the frequency spectrum, excluding the DC component which must remain constant. By contrast, since we model an unaltered image's histogram as an interpolatably connected function, we expect the histogram's DFT $H(k)$ to be a strongly low-pass signal. As a result, the presence of an appreciable amount of energy in the high frequency regions of $H(k)$ is indicative of contrast enhancement.

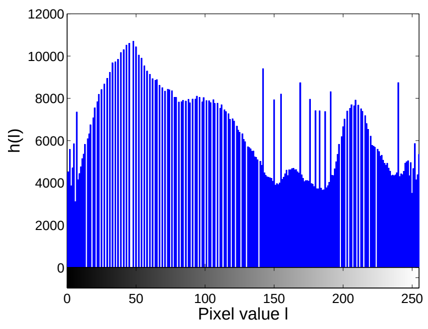
An alternate way of viewing this phenomena is to observe that locally contractive regions of a contrast enhancement mapping will cause multiple distinct input pixel values to be mapped to the same output pixel value. This will result in an isolated peak in the histogram of the contrast image at the common output pixel value. Similarly, locally expansive regions of a contrast enhancement mapping will cause adjacent input pixel values to be mapped apart, resulting in sudden gaps in the histogram of the enhanced image. Because these peaks and gaps are impulsive in nature, they will result in the presence of a significant high frequency component in $H(k)$. The bottom two plots of Fig. 2.4 show the frequency domain representations of the histogram of a typical image before and after it has undergone contrast enhancement.

Though an image's pixel value histogram is typically low-pass, this is not the case for an image whose histograms exhibit saturation effects. The impulsive component present in a saturated image's pixel value histogram will cause a DC offset to occur in its histogram's frequency domain representation which may be mistaken for the fingerprint of a contrast enhancement mapping. An example of this can be seen in Fig. 2.5, which shows a high end saturated image, its pixel value histogram, and the frequency domain representation of its histogram.

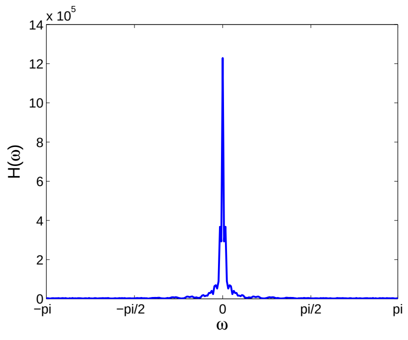
In light of these observations, we propose a technique which detects contrast enhancement by measuring the strength of the high frequency components of an image's pixel value histogram, then comparing this measurement to a predefined threshold. To prevent unaltered images exhibiting histogram saturation effects from yielding large high frequency measurements indicative of contrast enhancement map-



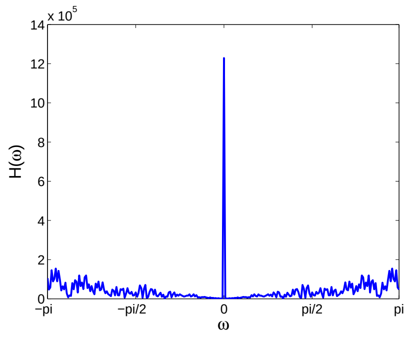
(a)



(b)



(c)



(d)

Figure 2.4: Pixel value histogram of (a) an unaltered image and (b) the same image after contrast enhancement has been performed, as well as the magnitude of the DFT of (c) the unaltered image's histogram and (d) the contrast enhanced image's histogram.

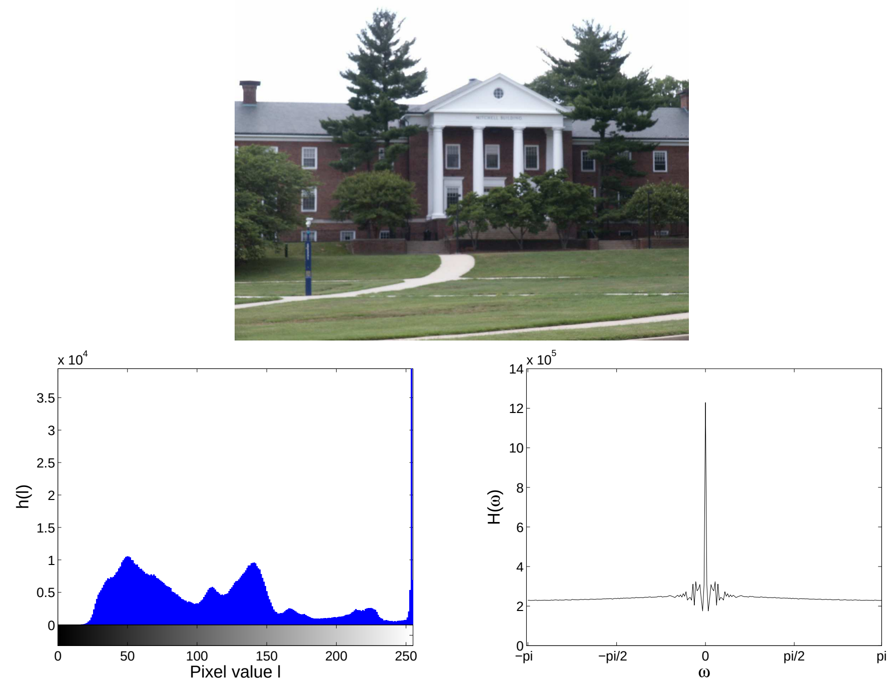


Figure 2.5: Top: Image exhibiting high end histogram saturation. Left: Histogram of the image's green pixel values. Right: Magnitude of the DFT of the image's green pixel value histogram.

ping fingerprints, we modify an image's histogram before testing so that it is free from saturation effects. This modified histogram $g(l)$ is obtained by performing the elementwise multiplication between $h(l)$ and a 'pinch off' function $p(l)$ so that

$$g(l) = p(l)h(l), \tag{2.11}$$

where

$$p(l) = \begin{cases} \frac{1}{2} - \frac{1}{2}\cos\left(\frac{\pi l}{N_p}\right) & l \leq N_p, \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi(l-255+N_p)}{N_p}\right) & l \geq 255 - N_p, \\ 1 & else, \end{cases} \tag{2.12}$$

and N_p is the width of the region over which $p(l)$ decays from 1 to 0. The pinch off function is designed to both remove impulsive histogram components which may occur due to saturation effects as well as to minimize the frequency domain effects of multiplying $h(l)$ by $p(l)$, which behaves similarly to a windowing function.

We calculate E, a normalized measure of the energy in the high frequency components of the pixel value histogram, from $g(l)$ according to the formula

$$E = \frac{1}{N} \sum_{k} |\beta(k)G(k)|, \tag{2.13}$$

where N is the total number of pixels in the image, $G(k)$ is the DFT of $g(l)$, and $\beta(l)$ is a weighting function which takes values between 0 and 1. The purpose of $\beta(l)$ is to deemphasize low frequency regions of $G(l)$ where nonzero values do not necessarily correspond to contrast enhancement artifacts. In this work, we use the simple cutoff function

$$\beta(k) = \begin{cases} 1 & c \leq k \leq 128, \\ 0 & else, \end{cases} \tag{2.14}$$

where c is the entry of the 256 point DFT corresponding to a desired cutoff frequency. $\beta(k)$ is zero for all values greater than $k = 128$ because symmetry properties inherent in the DFT of real valued signals make it unnecessary to measure these values.

After F has been calculated, the decision rule δ_{ce} is used to classify an image as unaltered or contrast enhanced, such that

$$\delta_{ce} = \begin{cases} \text{image is } \textit{not} \text{ contrast enhanced} & E < \eta_{ce}, \\ \text{image is contrast enhanced} & E \geq \eta_{ce}, \end{cases} \tag{2.15}$$

Our observation that an unaltered image's pixel value histogram is a strongly low-pass signal suggests that our detector's performance should improve as the frequency cutoff of c is increased. To verify this, we conducted an experiment on one set of data in which we obtained performance results for our contrast enhancement detection technique using c values ranging from 32 to 112 and compared the results. For this experiment, we used the green color layer from each of the 244 images in the Uncompressed Colour Image Database as a set of unaltered grayscale images [38]. We created a set of contrast enhanced images by applying the power law transformation

$$m(l) = 255\left(\frac{l}{255}\right)^{\gamma}, \tag{2.16}$$

with $\gamma = 1.1$ to the pixel values of each of the unaltered images. We then classified each of these images as altered or unaltered using a series of decision thresholds and with the parameter $N_p = 4$. The probabilities of detection P_d and false alarm P_{fa} were determined for each threshold by respectively calculating the percent of contrast enhanced images correctly classified and the percent of unaltered images incorrectly classified. The series of receiver operating characteristic (ROC) curves displayed in Fig. 2.6 was generated using these results. As we hypothesized, our detection algorithm's performance improved as the value of c was increased, with the best performance being achieved when using $c = 112$.

To perform a larger scale test of our contrast enhancement detection technique, we compiled a database of 341 unaltered images consisting of many different subjects and captured under a variety of light conditions. These images were taken with several different cameras and range in size from 1500×1000 pixels to 2592×1944 pixels. The green color layer of each of these images was used to create a set of unaltered grayscale images. We applied the power law transformation defined in

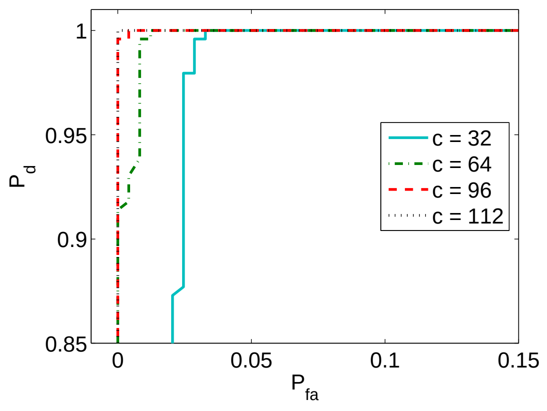
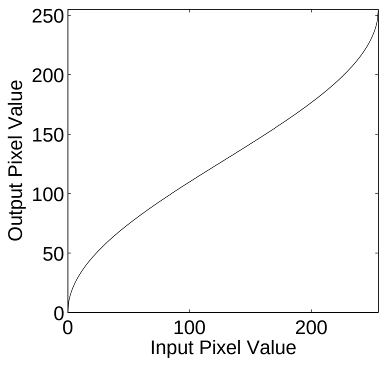


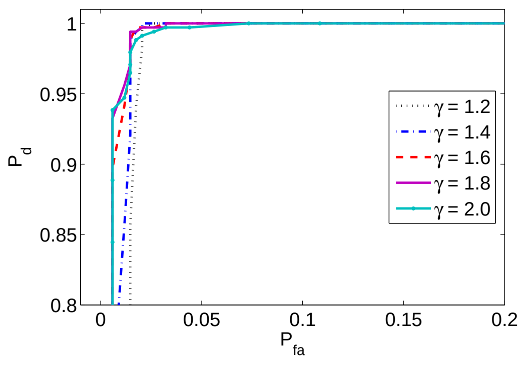
Figure 2.6: Contrast enhancement detection ROC curves for images altered by a power law transformation with (b) $\gamma = 1.1$ using several values of the cutoff parameter c.

(2.16) to each of these unaltered grayscale images using γ values ranging from 0.5 to 2.0 to create a set of contrast enhanced images. Additionally, we modified each unaltered grayscale image using the nonstandard contrast enhancement mapping displayed in Fig. 2.7(a). These images were combined with the unaltered images to create a testing database of 4092 grayscale images.

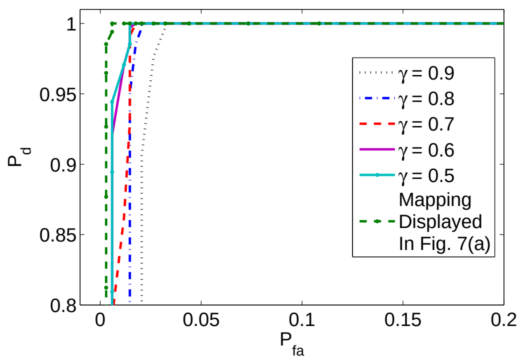
To evaluate the performance of our contrast enhancement detection technique on this testing set, each image was classified as altered or unaltered using a series of decision thresholds. During classification, the parameters N_p and c were set to $N_p = 4$ and $c = 112$. As before, the detection and false alarm probabilities were calculated at each decision threshold and the series of ROC curves shown in Figs. 2.7(b) and (c) were generated. For each form of contrast enhancement tested, our detection technique acheived a P_d of 0.99 at a P_{fa} of approximately 0.03 or less.



(a)



(b)



(c)

Figure 2.7: Contrast enhancement detection ROC curves for images altered by a power law transformation with (b) $2.0 \geq \gamma \geq 1.2$, and (c) $0.5 \geq \gamma \geq 0.9$ as well as the mapping displayed in Fig. (a).

2.3.2 Detection of Locally Applied Contrast Enhancement

Locally applied contrast enhancement can be defined as applying a contrast mapping to a set of contiguous pixels $\mathcal{J}$ within an image. If the cardinality of $\mathcal{J}$ is large enough that a histogram of the values of all pixels within $\mathcal{J}$ can be modeled as interpolatably connected, then when contrast enhancement is performed on the set $\mathcal{J}$ it will introduce its fingerprint into the histogram of $\mathcal{J}$. In light of this, the global contrast enhancement detection technique proposed in Section 2.3.1 can be performed on a test set of pixels $\mathcal{J}'$ to achieve localized contrast enhancement detection.

Ideally, the test set $\mathcal{J}'$ should be identical to the set $\mathcal{J}$ when performing localized contrast enhancement detection. In reality, this is seldom the case because if an image contains a set of contrast enhanced pixels, the members of this set are not public knowledge. In some scenarios, the authenticity of a particular image region is thrown in doubt and the test set can be manually selected to correspond to encompass this region. If localized contrast enhancement is carefully applied, however, it will not be obvious which image regions have been altered and the image must be searched for contrast enhancement in its entirety. This can be performed by segmenting an image into a set of blocks so that each block corresponds to a unique test set, then performing contrast enhancement detection on each block. The blockwise detection results can be combined to identify image regions which show signs of contrast enhancement.

In some scenarios, locally applied contrast enhancement detection can be used to identify other, more obviously malicious image manipulations such as cut-and-paste forgery. Cut-and-paste image forgery consists of creating a composite image by replacing a contiguous set of pixels in one image with a set of pixels $\mathcal{O}$ corresponding to an object from a separate image. If the two images used to create the composite image were captured under different lighting environments, an image forger may need to perform contrast enhancement on O so that lighting conditions match across the composite image. Failure to do this may result in a composite image which does not appear realistic. Image forgeries created in this manner can be identified by using localized contrast enhancement detection to locate $\mathcal{O}$, the cut-and-pasted region.

When performing blockwise localized contrast enhancement detection, it is important to ensure that the testing blocks are large enough to yield histograms suitable for contrast enhancement detection. If the blocks are too small, they may not contain enough pixels for the interpolatably connected histogram model to hold valid. In order to determine which block sizes are sufficient to perform reliable detection and examine the effectiveness of the local contrast enhancement detection scheme, we performed the following experiment. Each of the 341 unaltered images from the second test database described in Section 2.3.1 along with the power law transformed images corresponding to $\gamma = 0.5$ through 0.9 were segmented into square blocks. This process was performed for blocks of size 200×200, 100×100, 50×50, 25×25, and 20×20 pixels. Each block was then classified as contrast enhanced or unaltered by our contrast enhancement detection scheme using a variety of different thresholds. False alarm and detection probabilities were determined at each threshold and for every choice of block size by calculating the percent of incorrectly classified unaltered blocks and the percent of correctly classified contrast enhanced blocks respectively. This information was used to generate a set of ROC curves, shown in Fig. 2.8 for each value of γ which was tested.

The ROC curves shown in Fig. 2.8 indicate that local contrast enhancement can be reliably detected using testing blocks sized at least 100×100 pixels. At a P_{fa} of approximately 5%, a P_d of at least 95% was achieved using 200×200 pixel blocks and a P_d of at least 80% was achieved using 100×100 pixel blocks for each form of contrast enhancement tested. These results improved markedly when the contrast enhancement applied was stronger than the relatively mild power law transformation using $\gamma = 0.9$. In such cases, a P_d of roughly 98.5% and 96% was achieved with a P_{fa} of aproximatley 5% for blocks sized 200×200 pixels and 100×100 pixels respectively. It should also be noted that testing blocks sized 25×25 pixels and smaller appear to contain an insufficient number of pixels to perform reliable contrast enhancement detection.

An example of a cut-and-paste image forgery in which the pasted region has undergone contrast enhancement is shown in Fig. 2.9 along with the localized contrast enhancement detection results obtained from our proposed forensic technique. Adobe Photoshop was used to create the forged image shown in 2.9(c) from the unaltered images shown in Figs. 2.9(a) and 2.9(b). In order to detect the forgery, the image was then segmented into 100×100 pixel blocks, each of which was tested for evidence of locally applied contrast enhancement. Figs. 2.9(d)-(f) show the results of performing localized contrast enhancement detection on the red, green, and blue color layers of the composite image. Blocks corresponding to contrast enhancement

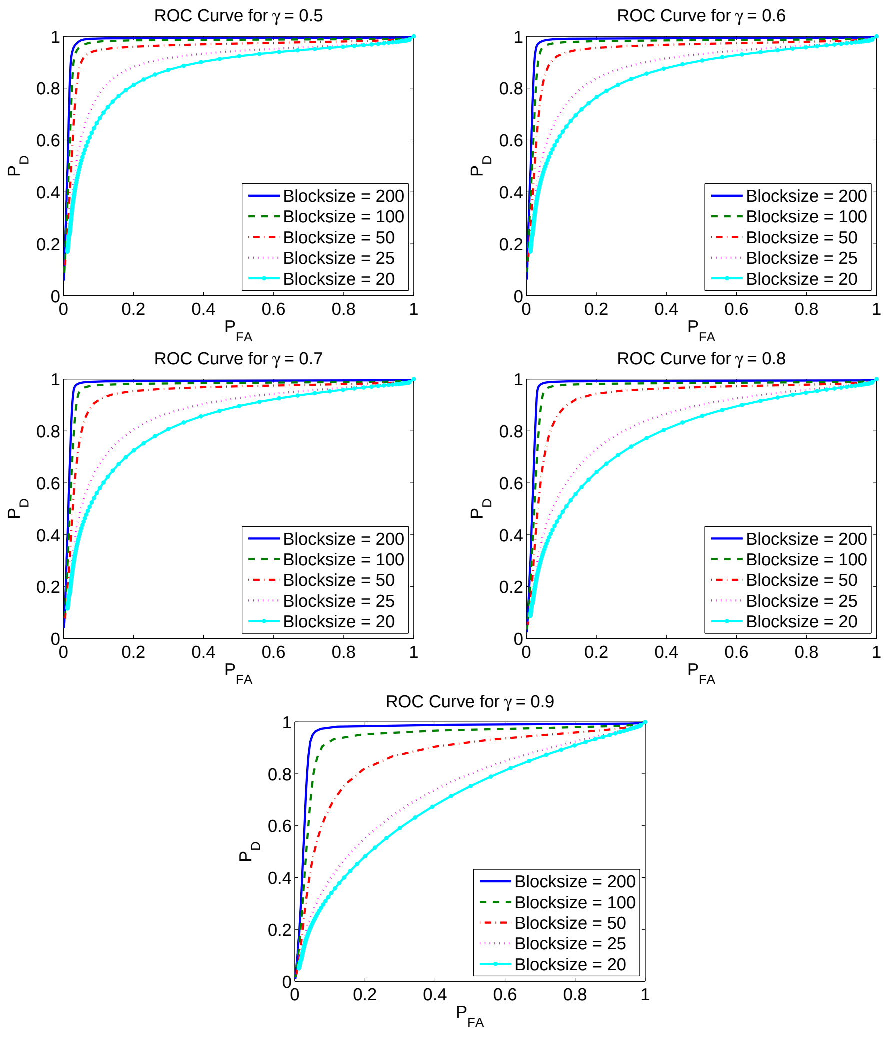


Figure 2.8: ROC curves obtained using different testing block sizes for images altered by a power law transformation with $\gamma = 0.5$ (top left), $\gamma = 0.6$ (top middle), $\gamma = 0.7$ (top right), $\gamma = 0.8$ (bottom left), and $\gamma = 0.9$ (bottom right).

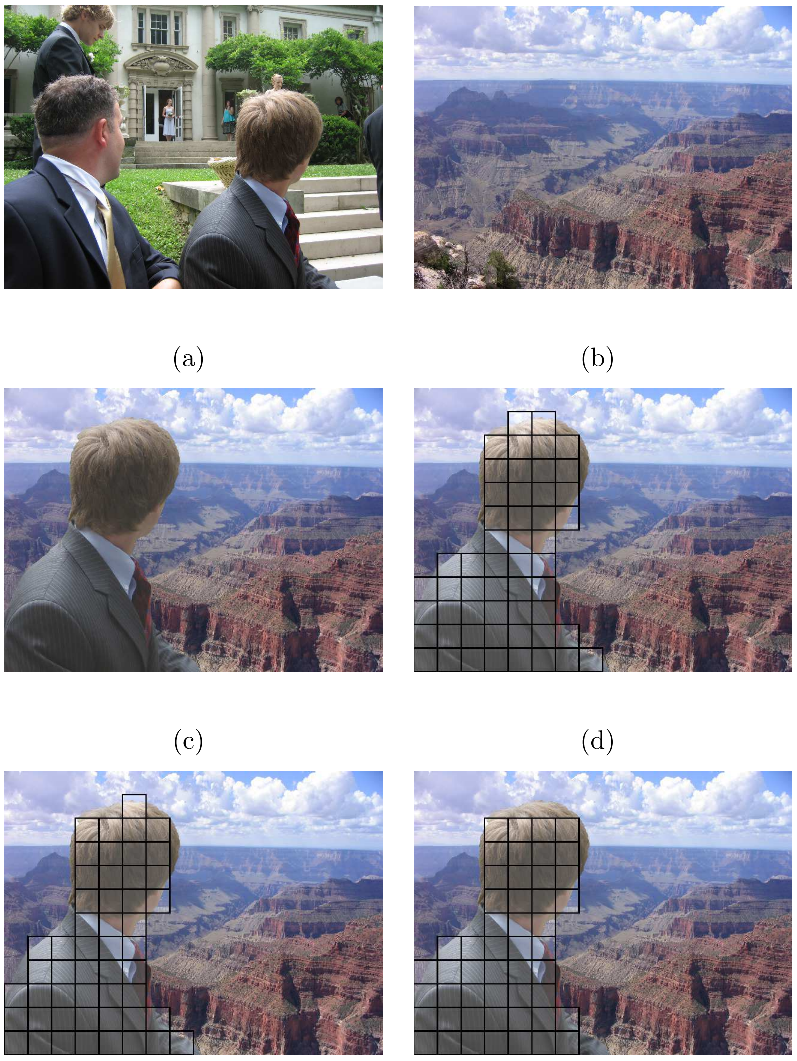

(a) (b) (c) (d) (e) (f)

Figure 2.9: Cut and paste forgery detection example showing (a) the unaltered image from which an object is cut, (b) the unaltered image into which the cut object is pasted, (c) the composite image, (d) red layer blockwise detections, (e) green layer blockwise detections, and (f) blue layer blockwise detections. Blocks detected as contrast enhanced are highlighted and boxed.

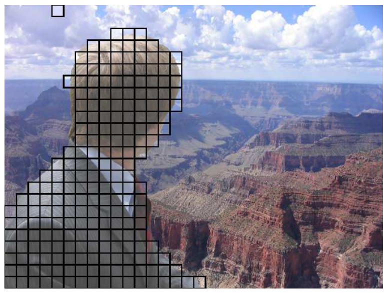
detections are highlighted and outlined in black. In this example, each of these blocks contain pixels that correspond to the inauthentic object.

Figure 2.10 shows detection results when the block size is reduced to 50×50 pixels. When detection was performed separately on each color layer of the forged image, several false alarms occurred, as can be seen in Figures 2.10(a)-(c). These false alarm blocks generally correspond to areas of the sky where the pixel values are nearly constant, leading to a pixel value histogram that contains an impulsive component outside of the pinch off region. The number of false alarms can be reduced in color images such as this by classifying a block as contrast enhanced only if a detection occurs at the corresponding block in each of the three color layers. Figure 2.10(d) shows the results of applying this detection criteria to the single color layer detections displayed in in Figures 2.10(a)-(c).

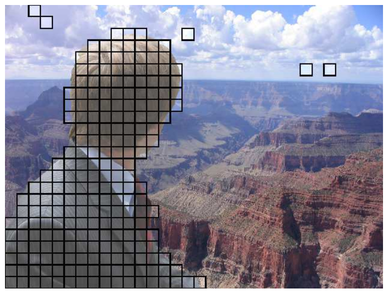
2.3.3 Histogram Equalization

In some scenarios, it may be desirable to identify the specific form contrast enhancement used to modify an image. One simple and commonly used form of contrast enhancement is histogram equalization. Histogram equalization effectively increases the dynamic range of an image's pixel values by subjecting them to a mapping such that the distribution of output pixel values is approximately uniform [14]. The mapping used to accomplish this is dependent upon the histogram of the unaltered image and is is generated according to the equation

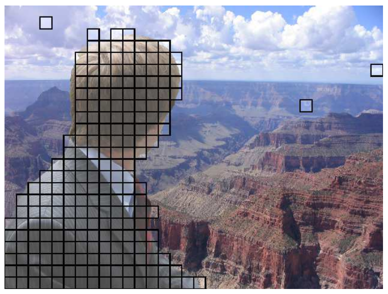
$$m_{he}(l) = \text{round}\left(255 \sum_{t=0}^{l} \frac{h(t)}{N}\right), \tag{2.17}$$



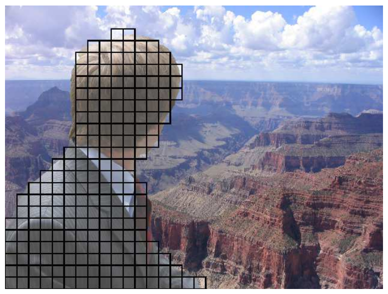
(a)



(b)



(c)



(d)

Figure 2.10: Cut and paste forgery detection results using 50×50 pixel blocks showing (a) red layer blockwise detections, (b) green layer blockwise detections, (c) blue layer blockwise detections, and (d) blockwise detections that occur across all three color layers.

where N is the total number of pixels in the image. Because the histogram of an unaltered image does not normally approximate a uniform distribution, the 'uniformity' of an equalized image's histogram can be used as an identifying feature of this mapping's intrinsic fingerprint. We propose a test which measures the distance between an image's normalized histogram and the uniform distribution, then uses this distance to determine if the image has undergone histogram equalization. This test can be used after contrast enhancement has been detected or it can performed independently of our generalized contrast enhancement detection technique.

Like any other contrast enhancement mapping, histogram equalization will introduce zeros into an image's pixel value histogram through the process discussed in Section 2.2. Because of this, measures such as the Kullback-Leibler divergence are ill equipped to determine the distance between the normalized histogram of an equalized image and the uniform distribution. Similarly, other measures such as the mean absolute difference or the mean squared difference between an image's normalized histogram and the uniform distribution will be biased away from small values indicative of a uniform histogram by the zeros and accompanying impulsive peaks present in an equalized image's histogram. To mitigate this problem, we propose measuring the uniformity of an image's histogram in the frequency domain, where histogram equalization's identifying features can be separated from other obfuscating effects.

The frequency domain representation of a constant function is an impulse centered at zero. Using this fact, we obtain a frequency domain measure of the distance D of an image's normalized histogram from the uniform distribution according to the formula

$$D = \frac{1}{N}\left(\sum_{k\neq 0} |H(k)|\alpha(k)\right), \tag{2.18}$$

In (2.18), $\alpha(k)$ is a weighting function used to deemphasize the high frequency regions in $H(k)$ where the energy introduced by histogram equalization's intrinsic fingerprint tends to accumulate. After calculating D for an image in question, the decision rule δ_{he} is then used to determine if histogram equalization has been performed, where

$$\delta_{he} = \begin{cases} \text{histogram equalization } \textit{not} \text{ present} & D > \eta_{he}, \\ \text{histogram equalization present} & D \leq \eta_{he}, \end{cases} \tag{2.19}$$

and η_{he} is the decision threshold.

As discussed in Section 2.3.1, frequency domain detection methods suffer problems due to the constant offset present in $H(k)$ in high and low end histogram saturated images. Multiplying $h(l)$ by a pinch off function will not remove the effects of histogram saturation because for histogram equalized images, the location of the impulsive component is often shifted by histogram equalization. Instead, we identify impulsive components which are likely due to saturation effects and remove them to obtain a modified histogram.

For low end histogram saturated images, we may safely assume that before histogram equalization is applied to an image, the impulsive nature of its histogram will cause the number of pixels in the lowest bin to be greater than $\frac{2N}{255}$. After histogram equalization is performed, the pixel value $l = 0$ will be mapped to an output value greater than or equal to 2 because

$$\begin{aligned} m_{he}(0) &= round\left(255\sum_{t=0}^{l=0}\frac{h(t)}{N}\right) \\ &\geq round\left(255\left(\frac{2}{255}\right)\right) = 2. \end{aligned} \tag{2.20}$$

Letting l' denote the lowest value of l such that $h(l) > 0$, images which may be of this nature can be identified if $l' \geq 2$ and $h(l') \geq \frac{2N}{255}$. For these images, the effects of the impulsive histogram component can be mitigated by forming a new histogram $h'(l)$ by retaining only the section of the histogram corresponding to pixel values larger than the k^{th} nonzero entry. More explicitly, $h'(l)$ can be defined as $h'(l) = h(l'_k + l + 1)$, where l'_k is the k^{th} nonempty bin in $h(l)$. The parameter $h(l)$ in (2.18) can then be replaced by $h'(l)$ to obtain a value of D unbiased by low end histogram saturations effects.

In the case of high end histogram saturated images, we can similarly assume that $h(255) \geq \frac{2N}{255}$. When histogram equalization is performed on these images, the input pixel value $l = 254$ is mapped to an output value of 253 or less because

$$\begin{aligned} m_{he}(254) &= round\left(255\sum_{t=0}^{l=254}\frac{h(t)}{N}\right) \\ &\leq round\left(255\left(1-\frac{2}{255}\right)\right) = 253. \end{aligned} \tag{2.21}$$

Using this information, high end saturated images that may have undergone histogram equalization can be identified by determining if $l'' \leq 253$ and $h(255) \geq \frac{2N}{255}$, where l'' is the largest value of l such that $l'' < 255$ and $h(l'') > 0$. A new histogram that does not contain the impulsive histogram component can now be formed by according by letting $h''(l) = h(l)$ for $l = 0, \ldots, l''_k - 1$, where l''_k is the k^{th} nonempty bin in $h(l)$ counting backwards from $l = 255$. As before, $h(l)$ in (2.18) can be replaced by $h''(l)$ to achieve a value of D unbiased by high end histogram saturations effects.

To evaluate the performance of our histogram equalization classification method, we performed histogram equalization on the 341 unaltered grayscale images from our global contrast enhancement test database described in Section 2.3.1. We combined the histogram equalized images with their unaltered counterparts to create a histogram equalization testing database. Next we used our detection algorithm to determine if each image in the database had undergone histogram equalization. Detection was performed using two different weighting functions,

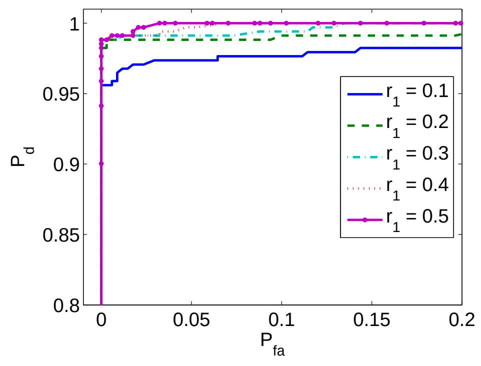
$$\alpha_1(k) = \begin{cases} exp(-r_1 k) & \text{if } 0 \leq k < 128 \\ exp(-r_1(256-k)) & \text{if } 128 \leq k \leq 255 \end{cases} \tag{2.22}$$

with r_1 taking values between 0.1 and 0.5 and

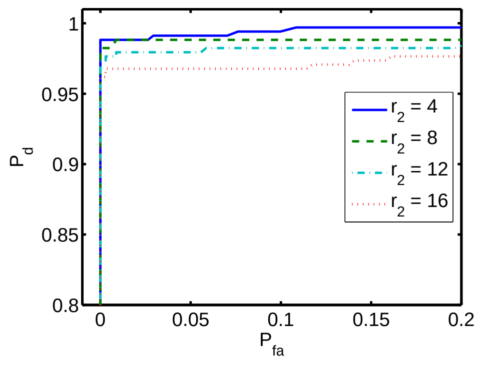
$$\alpha_2(k) = \begin{cases} 1 & \text{if } k \leq r_2 \text{ or } (256-k) \leq r \\ 0 & \text{else} \end{cases} \tag{2.23}$$

with r_2 values ranging from 4 to 16. The false alarm and detection probabilities were then determined by calculating the percentage of incorrectly classified unaltered images and the percentage of correctly classified histogram equalized images respectively.

A series of ROC curves showing the performance of our histogram equalization detection scheme are displayed in Figure 2.11. Our detector achieved its best performance using $\alpha_1(k)$ as a weighting function with $r_1 = 0.5$. Under these conditions, a P_d of 99% was reached with a P_{fa} of approximately 0.5% as well as a P_d of



(a)



(b)

Figure 2.11: Histogram equalization detection ROC curves obtained (a) using the weighting function defined in (2.22) and (b) using the weighting function defined in (2.23).

100% with a P_{fa} of nearly 3%. Additionally, Figure 2.11 shows that our detection scheme's performance improved as the value of r_1 increased when using $\alpha_1(k)$, and as the value of r_2 decreased when using $\alpha_2(k)$. Both of these trends correspond to an increase in detector performance as the weighting function is chosen to place more emphasis on low frequency regions of $H(k)$ during detection. This reinforces the notion that a weighting function is needed to deemphasize the middle and high frequency regions of $H(k)$ where general contrast enhancement artifacts can obscure evidence of histogram equalization.

Additionally, we performed an experiment to verify that our histogram equalization classification technique can differentiate between histogram equalization and other forms of contrast enhancement. We created a new testing database consisting of the 341 unaltered grayscale images as well as 1705 of the gamma corrected images corresponding to $\gamma = 0.5$ to 0.9 from the experiment discussed in Section 2.3.1. We then used our histogram equalization detection test to classify each of the images

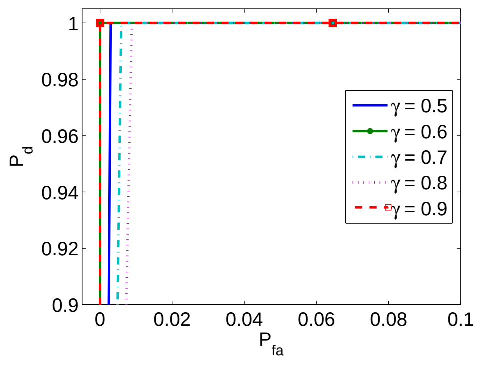


Figure 2.12: ROC curves obtained when differentiating between histogram equalization and other forms of contrast enhancement.

in the database as histogram equalized or not equalized. During classification, the weighting function described in (2.22) was used with $r_2 = 4$. The probabilities of detection and false alarm were obtained by calculating the percentage of correctly classified histogram equalized images and incorrectly classified gamma corrected images respectively. These probabilities were then used to generate the ROC curves displayed in Fig. 2.12. A P_d of 100% was achieved at a P_{fa} of less than 1% for each form of contrast enhancement tested.

2.4 Detecting Additive Noise in Previously JPEG Compressed Images

In this section, we present a technique designed to detect the global addition of noise to an image that has previously undergone JPEG compression. Though this may initially seem to be a fairly harmless operation, additive noise can be used to disguise visual traces of image forgery or in an attempt to mask statistical artifacts left behind by other image altering operations. Previous work has dealt with the detection of noise added to specific regions of an image by searching for fluctuations in localized estimates of an image's signal to noise ratio (SNR) [33]. This method fails, however, when noise has been globally added to an image because this scenario will not result in localized SNR variations. Instead of relying upon SNR measurements, our proposed technique operates by applying a predefined mapping with a known fingerprint to a potentially altered image's pixel values [43]. This mapping is chosen such that an identifying feature of its fingerprint will be absent if noise was added to the image. Accordingly, we are able to detect the presence of additive noise if the application of the predefined mapping does not introduce a fingerprint with this feature.

2.4.1 Scale and Round Mapping

To perform additive noise detection, we make use of a mapping which we refer to as the *scale and round* mapping. We define the scale and round mapping as

$$v = \text{round}(cu) \tag{2.24}$$

where $u, v \in \mathbb{Z}$ and c is a fixed scalar. To understand the fingerprint left by this mapping, let us also define $\mathcal{U}_c(v)$ as the set of u values mapped to each distinct v value by (2.24), where

$$\mathcal{U}_c(v) = \{u | v = \text{round}(cu)\}. \tag{2.25}$$

The cardinality of this set, denoted by $|\mathcal{U}_c(v)|$, depends on the values of both c and v. It can be proven that if $c = \frac{p}{q}$ such that $p, q \in \mathbb{Z}$ are relatively prime, $|\mathcal{U}_c(v)|$ is periodic in v with period p. To see why this is so, consider first the following two easily proven lemmas:

Lemma 1: Given $a \in \mathbb{Z}$ and $b \in \mathbb{R}$

$$a = \text{round}(b) \Leftrightarrow a + k = \text{round}(b + k), \forall k \in \mathbb{Z}. \tag{2.26}$$

Lemma 2: Given $u, v \in \mathbb{Z}$ and $c = \frac{p}{q}$ such that $p, q \in \mathbb{Z}$ are relatively prime

$$v = \text{round}(cu) \Leftrightarrow v + p = \text{round}(c(u + q)) \tag{2.27}$$

Now using Lemma 2, we can state that for all $u \in \mathcal{U}_c(v)$, there exists some $\tilde{u} \in \mathcal{U}_c(v + p)$, namely $\tilde{u} = u + q$, which implies that $|\mathcal{U}_c(v)| = |\mathcal{U}_c(v + p)|$. This proves that the number of u values mapped to each v value is periodic with period p. As a consequence, the intrinsic fingerprint of the scale and round operation will contain a periodic component with period p.

2.4.2 Hypothesis Testing Scenario

We now shift our discussion to JPEG compression and its significance to the detection of additive noise. When a color image undergoes JPEG compression, each pixel in the image is first converted from the RGB color space to the YCbCr color space using a linear transformation. Next, each color layer is divided into a series of 8×8 pixel blocks and the discrete cosine transform of each block is computed. The resulting set of DCT coefficients are quantized by dividing each coefficient by its corresponding entry in a quantization matrix, then rounding the result to the nearest integer. Finally, the quantized DCT coefficients are reordered into a single bitstream which is losslessly compressed.

The image is decompressed by losslessly decoding the bitstream of quantized DCT coefficients, then reshaping it back into the series of blocks. The DCT coefficients are dequantized by multiplying each quantized DCT coefficient by its corresponding entry in the quantization matrix used during compression. Next, the inverse DCT (IDCT) of each block is computed, resulting in a set of pixel values in the YCbCr colorspace. Because the dequantized DCT coefficients are integer multiples of their respective quantization table entries and because the IDCT is a fixed linear transformation, the pixel values in the YCbCr color space will lie in a countable subset of $\mathbb{R}^3$. As a result, if a monotonically increasing mapping is applied to any color layer in the YCbCr color space, that mapping's fingerprint will be introduced into the histogram of the color layer's values.

In the final stage of JPEG decompression, the pixels are transformed from the YCbCr to the RGB color space, then projected back into $\mathcal{P}^3$. Letting $\mathbf{y}$ denote a pixel in the RGB color space, $\mathbf{x}$ denote the same pixel in the YCbCr color space, and $\mathbf{T}$ be the linear transformation that maps a pixel from the YCbCr to the RGB color space, this process can be described mathematically by the equation

$$\mathbf{y} = \text{truncate}(\text{round}(\mathbf{Tx})), \tag{2.28}$$

where the operation truncate($\cdot$) maps values of its argument less than 0 to 0 and values greater than 255 to 255. By defining $Q(\mathbf{Tx}) = \text{truncate}(\text{round}(\mathbf{Tx})) - \mathbf{Tx}$, we may now formulate the detection of additive noise as the following hypothesis testing problem:

$$\begin{aligned} H_0 &: \mathbf{y} = \mathbf{T}\mathbf{x} + Q(\mathbf{T}\mathbf{x}) \\ H_1 &: \mathbf{y} = \mathbf{T}\mathbf{x} + Q(\mathbf{T}\mathbf{x}) + \mathbf{n}. \end{aligned} \tag{2.29}$$

It should be noted that traditional Bayesian techniques cannot be used to differentiate between these two hypotheses because the distribution of $\mathbf{x}$ is unknown. Instead, we differentiate between these two hypotheses by observing that the fingerprint left by the mapping

$$\mathbf{z} = \text{round}(c\mathbf{T}^{-1}\mathbf{y}), \tag{2.30}$$

where the constant $c = \frac{p}{q}$ is such that $p, q \in \mathbb{Z}$ are relatively prime, differs under each hypothesis. When this mapping is applied to each pixel within an image, the hypothesis testing problem outlined in (2.29) can be rewritten as

$$\begin{aligned} H_0 &: \mathbf{z} = \text{round}(c\mathbf{x} + \mathbf{e}) \\ H_1 &: \mathbf{z} = \text{round}(c\mathbf{x} + \mathbf{e} + c\mathbf{T}^{-1}\mathbf{n}), \end{aligned} \tag{2.31}$$

where $\mathbf{e} = c\mathbf{T}^{-1}Q(\mathbf{T}\mathbf{x})$.

Under hypothesis H_0, the i^{th} entry of $\mathbf{z}$ can be expressed as according to the formula

$$\begin{aligned} z_i &= \text{round}(cx_i + e_i) \\ &= \text{round}(cx_i) + \text{round}(e_i) + d_i, \end{aligned} \tag{2.32}$$

where d_i is an independent random variable which accounts for the error induced by summing the individually rounded terms cx_i and e_i. Because the variances of the terms round(e_i) and d_i are typically small, the term round(cx_i) dominates the behavior of the PMF of z_i. Since the term round(cx_i) is of the same form as (2.24), the number of distinct x_i values mapped to each z_i value will occur in a fixed periodic pattern. This will result in the presence of a discernible periodic pattern with period p in the envelope of the histogram of z_i values. This pattern corresponds to the intrinsic fingerprint of the scale and round mapping. This pattern, which corresponds to the intrinsic fingerprint of the scale and round mapping, can be clearly seen in Fig. 2.13.

Under hypothesis H_1, we find that the histogram of z_i values exhibits different behavior. Defining the matrix $\mathbf{W}$ as the inverse of $\mathbf{T}$ such that

$$\mathbf{T}^{-1} = \mathbf{W} = \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix}, \tag{2.33}$$

the i^{th} entry of $\mathbf{z}$ can be expressed as

$$\begin{aligned} z_i &= \text{round}\Big(cx_i + \sum_{j=1}^{3} cW_{i,j}n_j + e_i\Big) \\ &= \text{round}(cx_i) + \sum_{j=1}^{3} \text{round}\left(cW_{i,j}n_j\right) + \text{round}(e_i) + d_i, \end{aligned} \tag{2.34}$$

where d_i is an independent random variable which accounts for the error induced by moving the summation of terms outside the round operation. Under this hypothesis, the PMF of z_i is equivalent to the convolution of each of these terms. Under this hypothesis, however, three additional terms containing the scale and round mapping appear, each with their own scaling constant $cW_{i,j}$. If these scaling constants along with the original scaling constant c are such that the fingerprints introduced into each individual term share no common period, then the convolution of the PMFs of each term will effectively smooth out the PMF of z_i. As a result, no periodic pattern will be introduced into the histogram of z_i by the mapping defined in (2.28). This effect can be observed in the example shown in Fig. 2.13.

2.4.3 Additive Noise Detection in Images

Using this information, we are able to rephrase the detection of the addition of noise to a previously JPEG compressed image as the detection of the periodic fingerprint of (2.28) within the envelope of $h_{z_i}(l)$, the normalized histogram of z_i. Because of its periodic nature, the detection of this fingerprint is particularly well suited for the frequency domain, where it will produce a peak centered at the frequency bin corresponding to its fundamental frequency or an integer multiple thereof. The bottom two plots of Fig. 2.13 show the presence or absence of this peak under each hypothesis. Furthermore, since the period of the fingerprint is dictated by our choice of the scaling constant, we are able to choose the frequency location of this peak.

To facilitate detection, we obtain a frequency domain representation $G_{z_i}(k)$ of the histogram $h_{z_i}(l)$ which is free from any possible high or low end histogram saturation effects. We accomplish this by defining $G_{z_i}(k)$ as the DFT of $g_{z_i}(l)$, which we calculate using the equation

$$g_{z_i}(l) = h_{z_i}(l)p(l) \tag{2.35}$$

where $p(l)$ is the pinch off function denoted in (2.12). Next, we test for the presence of the periodic fingerprint by measuring the strength of the peak that it introduces into $G_{z_i}(k)$. This measurement is obtained using the equation

$$S = \min\left\{ \frac{|G_{z_i}(k^*)|}{\frac{1}{|\mathcal{B}_1|}\sum_{j\in\mathcal{B}_1}|G_{z_i}(j)|}, \frac{|G_{z_i}(k^*)|}{\frac{1}{|\mathcal{B}_2|}\sum_{j\in\mathcal{B}_2}|G_{z_i}(j)|} \right\} \tag{2.36}$$

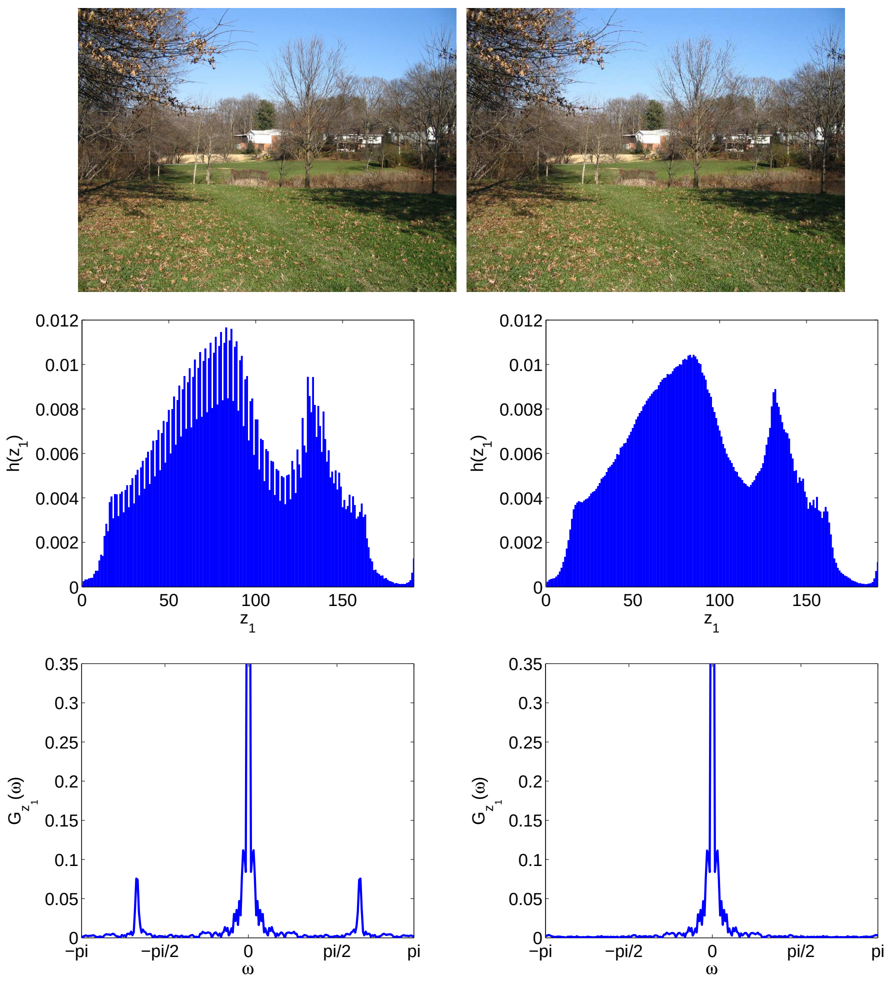


Figure 2.13: Example showing an unaltered image (top left), its normalized z_1 histogram (middle left), and the magnitude of the DFT of its z_1 histogram (bottom left), as well as an altered version of the image to which unit variance Gaussian noise has been added (top right), its normalized z_1 histogram (middle right), and the magnitude of the DFT of its z_1 histogram (bottom right). In both cases, the scaling parameter was chosen to be $c = \frac{3}{4}$.

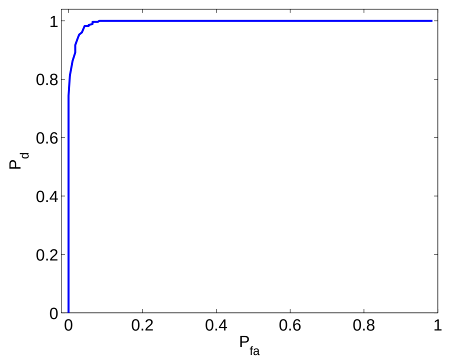


Figure 2.14: Additive noise detection ROC curve for images which were JPEG compressed using default camera settings then altered by adding unit variance Gaussian additive noise.

where k^* is the frequency location of the expected peak and $\mathcal{B}_1$ and $\mathcal{B}_2$ are sets of contiguous indices of G_{z_i} lying above and below k^* respectively. Finally, we use a decision rule δ_n corresponding to the threshold test

$$\delta_n = \begin{cases} \text{noise has } \textit{not} \text{ been added} & \text{if } S < \eta_n \\ \text{noise has been added} & \text{if } S \geq \eta_n. \end{cases} \tag{2.37}$$

to determine the presence or absence of additive noise within the image.

When using this technique, the sets $\mathcal{B}_1$ and $\mathcal{B}_2$ should be chosen such that they do not include indices directly adjacent to k^*. This is because DFT windowing effects may result in artificially larger values of $|G_{z_i}(k)|$ around the peak if it is present. Additionally, the interpolatable connectivity restriction placed upon the histogram of pixel values in our image model implies that $G_{z_i}(k)$ will be strongly low-pass in nature. This property suggests that to achieve better differentiability, c should be chosen such that it introduces a high frequency signal into $h_{z_i}(l)$.

To evaluate the performance of our additive noise detection technique, we compiled a set of 277 unaltered images taken by four different digital cameras from unique manufacturers. These images capture a variety of different scenes and were saved as JPEG compressed images using each camera's default settings. A set of altered images was created by decompressing each image and independently adding unit variance Gaussian noise to each pixel value. These altered images were then saved as bitmaps, along with decompressed versions of the original images, creating a testing database of 554 images. Next we used our additive noise detection test to determine if noise had been added to each image in the database. When creating the histogram of z_i values, we chose $i = 1$ which corresponds to using the luminance or "Y" component of each pixel. The parameter c was chosen to take the value $c = \frac{7}{11}$ leading to an expected peak location of $k^* = 71$. The sets of $\mathcal{B}_1$ and $\mathcal{B}_2$ were chosen to be $\mathcal{B}_1 = \{61, \ldots, 68\}$ and $\mathcal{B}_2 = \{74, \ldots, 81\}$.

Detection and false alarm probabilities were determined at a series of decision thresholds by calculating the percentages of correctly classified images to which noise had been added and incorrectly classified unaltered images respectively. Using this data, an ROC curve showing the performance of our additive noise detection algorithm is displayed in Fig. 2.14. A P_d of approximately 80% was achieved at a false alarm rate less than 0.4%. When the P_{fa} was held less than 6.5%, the P_d increased to nearly 99%. These results indicate that our detection scheme is able to reliably detect additive noise in images previously JPEG compressed using a camera's default settings.

Additionally, we evaluated our additive noise detection technique's ability to operate on images previously JPEG compressed at different quality factors. To do this, we JPEG compressed each of the 244 images in the Uncompressed Colour

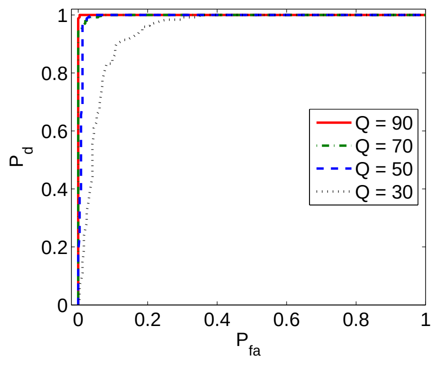


Figure 2.15: Additive noise detection ROC curve for images which were JPEG compressed at several different quality factors then altered by adding unit variance Gaussian additive noise.

Image Database at the quality factors $Q = 90$, 70, 50, and 30 [38]. As before, we created a set of altered images by adding unit variance Gaussian noise to each image, then saved each image as a bitmap. We then tested each image for the presence of additive noise with our proposed forensic technique using a variety of detection thresholds. We conducted this experiment using the same experimental parameters as our previous test. For each threshold, the probabilities of detection and false alarm were calculated then used to construct the series of ROC curves displayed in Fig. 2.15. Results comparable to our previous experiment were achieved for images previously compressed using quality factors of 50 or greater. For these quality factors, a P_d of 99% was acheived at a P_{fa} of 3.7% or less. At lower quality factors, however, noise detection appears to become more difficult.

2.5 Summary

In this chapter, we proposed a set of digital image forensic techniques capable of detecting global and local contrast enhancement, identifying the use of histogram equalization, and detection the global addition of noise to a previously JPEG compressed image. In each of these techniques, detection depends upon the presence or absence of an intrinsic fingerprint introduced into an image's histogram by a pixel value mapping.

We developed a model of an unaltered image's pixel value histogram and provided justification for this model. We defined the intrinsic fingerprint which a mapping leaves in the histogram of of an image's pixel values or other discrete valued data. By observing that the intrinsic fingerprints of contrast enhancement operations add energy to the high frequency components of an image's pixel value histogram, we developed a global contrast enhancement detection technique. We extended this technique into a method for detecting locally applied contrast enhancement and demonstrated its usefulness for detecting cut and paste type forgeries. Characteristic features of histogram equalization's intrinsic fingerprint were identified and used to propose a scheme for identifying the use of this operation. Additionally, we proposed a technique which detects the global addition of noise to a previously JPEG compressed image by searching for the intrinsic fingerprint of a specific pixel value mapping applied to the image in question.

Through detailed simulations, we tested the effectiveness of each of the proposed forensic techniques. Our simulation results show that aside from exceptional cases, each of the proposed techniques achieved a P_d of 99% with a P_{fa} of 7% or less. These results indicate that all of the proposed forensic techniques are very useful tools for identifying image manipulations and forgeries.

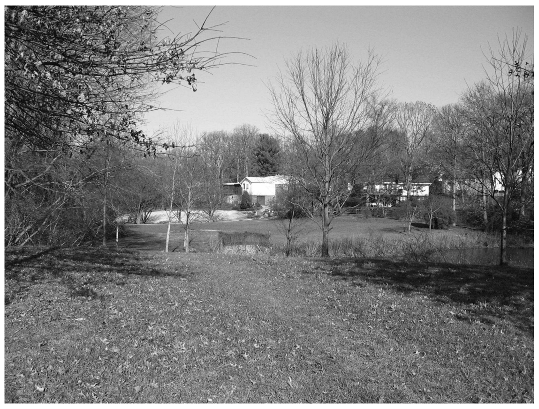
Chapter 3

Forensic Estimation and Reconstruction of a Contrast Enhancement Mapping

In the previous chapter, we identified the intrinsic fingerprints of contrast enhancement operations and used them to identify contrast enhanced images [40]. In this chapter, we present an iterative method to jointly estimate the contrast enhancement mapping used to modify an image as well as the image's pixel value histogram before contrast enhancement [45]. Our method requires no side information and makes no assumptions on the form of the contrast enhancement mapping aside from monotonicity. This algorithm is more general than previous work such as [8], which assumes that the contrast enhancement mapping can be described by a parametric equation which is known to the forensic examiner.

3.1 System Model

For any digital image, a normalized histogram h of its pixel values can be computed such that each normalized histogram value $h(x) \in [0, 1]$ represents the relative frequency of a pixel value x. Because most images vary in their origin and content, their histogram statistics will vary as well. For images created by using a digital camera to capture a real world scene, however, we have observed that their pixel value histograms typically conform to a smooth envelope, as can be seen in



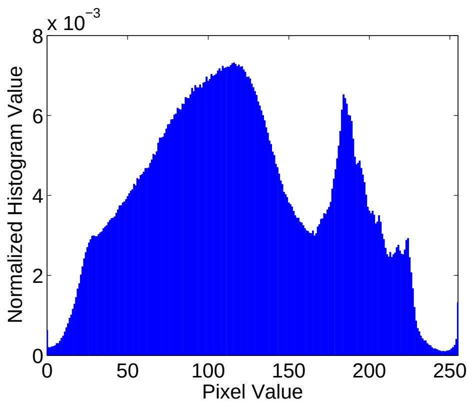


Figure 3.1: Left: Typical image captured by a digital camera. Right: Pixel value histogram of the image shown above.

Fig. 3.1. This phenomenon is caused by many factors, including complex lighting and shading environments, electronic noise present in a digital camera's CCD sensor, and the observation that most real world scenes consist of a continuum of colors [40].

Because no notion of smoothness exists for discrete functions such as histograms, we instead describe the pixel value histograms of these image as *interpolatably connected*. We define an interpolatably connected histogram as one in which an approximation of the histogram value at a particular pixel value x' can be obtained by interpolating $h(x')$ given all other histogram values using a smoothing spline. We denote this approximated value as $\hat{h}(x')$. Fig. 3.2 shows an approximation of the pixel value histogram displayed in Fig. 3.1, where each approximated value has been calculated in the manner described above. Though the histograms of computer generated images may not have this property, it is unlikely that such images would undergo contrast enhancement since their contrast properties can be controlled during the image's creation. As a result, we only consider images captured using a digital camera in this work.

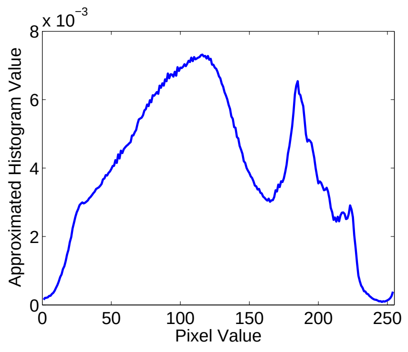


Figure 3.2: Approximation $\hat{h}$ of the histogram shown in Fig. 3.1.

We model the relationship between a histogram value and its smoothing spline approximation using the formula

$$h(x) = (1 + \epsilon)\hat{h}(x), \tag{3.1}$$

where the term ϵ is a random variable which takes into account approximation error. We choose a multiplicative error model instead of an additive one to account for the fact that large differences between h and $\hat{h}$ are more probable for large values of $\hat{h}$. After observing the distribution of ϵ values, which can be calculated from a histogram and its approximation using the formula $\epsilon = \frac{h(x)-\hat{h}(x)}{\hat{h}(x)}$, we model the distribution of the multiplicative error term as

$$P(\epsilon = q) = \tfrac{\lambda}{2-e^{-\lambda}} e^{-\lambda|q|}\, \mathbb{1}(q \geq -1) \tag{3.2}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function. The validity of this model distribution can be seen in Fig. 3.3, which shows the distribution of ϵ values calculated from the histogram and histogram approximation shown in Figs. 3.1 and 3.2 respectively, as well as the fitted model distribution.

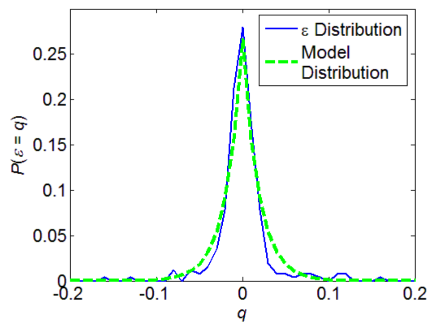


Figure 3.3: Distribution of ϵ values calculated from the histogram h shown in Fig. 3.1 and its approximation $\hat{h}$ shown in Fig. 3.2

3.2 Effects of Contrast Enhancement

When a contrast enhancement operation is applied to a digital image, its pixel values undergo a nonlinear mapping. Letting $\mathbb{P} = \{0, \ldots, 255\}$ denote the set of allowable pixel values, each pixel value $x \in \mathbb{P}$ in the unaltered image is mapped to a pixel value $y \in \mathbb{P}$ in the contrast enhanced image using the mapping function m, such that

$$y = m(x). \tag{3.3}$$

To exclude simple reorderings of the pixel values, we assume that m is monotonically nondecreasing.

Because contrast enhancement alters the pixel values of an image, its pixel value histogram will be affected as well. The histogram $h_Y(y)$ of pixel values in the contrast enhanced image can be written in terms of the unaltered image's pixel value histogram $h_X(x)$ using the equation

$$h_Y(y) = \sum_{x \in \mathbb{P}} h_X(x)\, \mathbb{1}(m(x) = y). \tag{3.4}$$

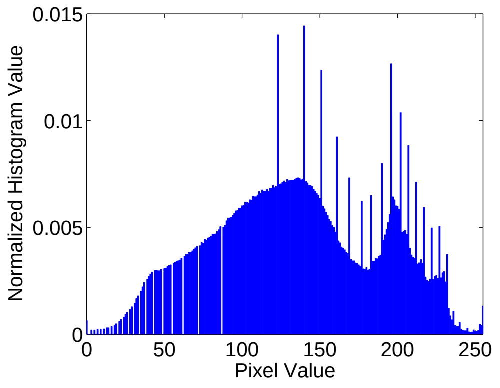


Figure 3.4: Pixel value histogram of the image shown in Fig. 3.1 after contrast enhancement has been applied.

This equation indicates that every value of h_Y must equal either a single value of h_X, a sum of distinct h_X values, or zero. As a consequence, impulsive peaks will occur in h_Y at y values to which multiple x values were mapped. Similarly, gaps will occur in h_Y at y values to which no x values were mapped. These peaks and gaps, which can be clearly seen in Figure 3.4, serve as the contrast enhancement fingerprints discussed in Chapter 2 that can be used to identify contrast enhancement [40].

3.3 Estimation of the Contrast Enhancement Mapping and the Unaltered Histogram

Once an image has been identified as contrast enhanced, an estimate of the contrast enhancement mapping used to modify the image as well as an estimate of the unaltered image's pixel value histogram can be jointly obtained through an iterative process. In this section, we describe this iterative process in detail. To aid the reader, we have included Fig. 3.5 which shows an example of our proposed algorithm over multiple iterations. The histogram entries in this example have been uniquely color coded so that they can be tracked across each iteration. When multiple histogram entries share a common color in iterations 1 and 2, it is because the estimate of the contrast enhancement mapping at that iteration indicates that their corresponding pixel values are mapped to the same contrast enhanced pixel value.

We define $g^{(k)}(x)$ as the k^{th} estimate of the unaltered image's histogram. This estimate is initialized by setting $g^{(0)}$ equal to contrast enhanced image's histogram h_Y. Each iteration begins by searching for the entry in $g^{(k)}(x)$ most likely to correspond to the sum of multiple entries of the unaltered image's histogram h_X. This is equivalent to finding the pixel value $x_*^{(k)}$ present in the contrast enhanced image that is most likely to be one to which multiple unaltered pixel values were mapped. To do this, we establish a test set of pixel values that could potentially be $x_*^{(k)}$. Because pixel values whose estimated histogram value are zero cannot be $x_*^{(k)}$, nor can any pixel that maps to a past value of x_*, we define the test set at the k^{th} iteration as $T^{(k)} = \{x | x \in \mathbb{P}, g^{(k)}(x) \neq 0, x \notin X^{(k)}\}$ where $X^{(k)}$ is the set of pixel values that map to previous values of x_*. The set $X^{(k)}$ is initialized as the null set and an update rule for $X^{(k)}$ is given later in (3.11).

Since finding $x_*^{(k)}$ is equivalent to finding the entry of $g^{(k)}$ least likely to correspond to a single entry in the unaltered histogram h_X, we may rephrase our problem as

$$x_*^{(k)} = \arg \min_{x \in T^{(k)}} P(h_X(x) = g^{(k)}(x)). \tag{3.5}$$

If we temporarily assume that the histogram approximation $\hat{h}$ is known, we may use (3.1) and (3.2) to write

$$\begin{aligned} P\left(h_X(x) = g^{(k)}(x)\right) &= P\left(\epsilon = \tfrac{g^{(k)}(x)-\hat{h}(x)}{\hat{h}(x)}\right) \\ &= \tfrac{\lambda}{2-e^{-\lambda}} e^{-\lambda\left|\frac{g^{(k)}(x)-\hat{h}(x)}{\hat{h}(x)}\right|} \mathbb{1}\left(\tfrac{g^{(k)}(x)-\hat{h}(x)}{\hat{h}(x)} \geq -1\right), \end{aligned} \quad (3.6)$$

therefore $x_*^{(k)}$ will be the value of $x \in T^{(k)}$ which maximizes $\left|\frac{g^{(k)}(x)-\hat{h}(x)}{\hat{h}(x)}\right|$ such that $\frac{g^{(k)}(x)-\hat{h}(x)}{\hat{h}(x)} \geq -1$. Since it is unlikely that $\hat{h}$ is known, we approximate it with $\hat{g}^{(k)}$, where $\hat{g}^{(k)}(x)$ is determined by using a smoothing spline to interpolate the value of $g^{(k)}(x)$ given $\{g^{(k)}(x')|x' \in T, x' \neq x\}$. Using this approximation, the value of $x_*^{(k)}$ is determined according to the formula

$$x_*^{(k)} \approx \arg\max_{x \in T^{(k)}} \left|\tfrac{g^{(k)}(x)-\hat{g}^{(k)}(x)}{\hat{g}^{(k)}(x)}\right| \mathbb{1}\left(\tfrac{g^{(k)}(x)-\hat{g}^{(k)}(x)}{\hat{g}^{(k)}(x)} \geq -1\right). \quad (3.7)$$

The function to be maximized in (3.7) is nonconcave, therefore $x_*^{(k)}$ must be found using an exhaustive search. Fortunately, this search is not prohibitively time consuming due to the relatively small number of elements in $T^{(k)}$.

Once $x_*^{(k)}$ has been determined, the estimate of the contrast enhancement mapping, denoted by $m^{(k)}$, can be updated. Before the first iteration, $m^{(0)}$ is initialized such that $m^{(0)}(x) = x$. To update $m^{(k)}$ we first estimate r, the number of unaltered pixel values mapped to $x_*^{(k)}$. By assuming that the histogram value of each pixel value mapped to $x_*^{(k)}$ is $\hat{g}^{(k)}(x_*^{(k)})$, we can obtain r using the equation

$$r = \text{round}\left(\frac{g^{(k)}(x_*^{(k)})}{\hat{g}^{(k)}(x_*^{(k)})}\right). \quad (3.8)$$

Because the contrast enhancement mapping is monotonically nondecreasing, it must preserve the pixel value ordering. This implies that the set of pixel values mapped

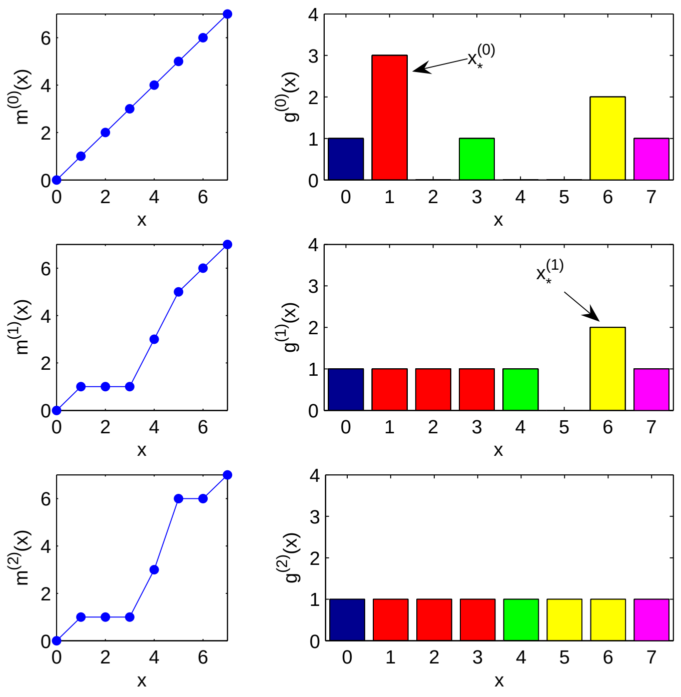


Figure 3.5: Example of our estimation algorithm running across several iterations. Histogram entries are initially uniquely color coded so that they can be tracked across each iteration. Histogram entries share a common color in iterations 1 and 2 when the current contrast enhancement estimate indicates that their corresponding pixel values will be mapped to the same output pixel value. The histogram values of these entries are estimated in accordance with our algorithm.

to $x_*^{(k)}$ must lie either immediately above or below $x_*^{(k)}$. Furthermore, since a zero is introduced somewhere into the pixel value histogram by each pixel value mapped to $x_*^{(k)}$, we determine which pixel values were mapped to $x_*^{(k)}$ by counting the number of zeros in $g^{(k)}$ both above and below $x_*^{(k)}$. We define these counts as $n_+ = \sum_{x > x_*^{(k)}} \mathbb{1}(g^{(k)}(x) = 0)$ and $n_- = \sum_{x < x_*^{(k)}} \mathbb{1}(g^{(k)}(x) = 0)$.

If $n_+ \geq n_-$, we assume that the r pixel values immediately greater than $x_*^{(k)}$ are mapped to $x_*^{(k)}$ and that all pixel values greater than $x_*^{(k)} + r$ are shifted by the mapping accordingly. Precisely, we update the mapping in this case according to the equation

$$m^{(k+1)}(x) = \begin{cases} m^{(k)}(x) & x < x_*^{(k)} \\ x_*^{(k)} & x_*^{(k)} \leq x < x_*^{(k)} + r \\ m^{(k)}\left(x + \sum_{l=x}^{x_+} \mathbb{1}(g^{(k)}(l) = 0)\right) & x_*^{(k)} + r \leq x \leq x_+ \\ m^{(k)}(x) & x > x_+ \end{cases} \tag{3.9}$$

where x_+ is the location of the r^{th} zero in $g^{(k)}$ counting up from $x_*^{(k)}$. Similarly, if $n_+ < n_-$ we assume that the r pixel values immediately less than $x_*^{(k)}$ are mapped to $x_*^{(k)}$ and that the pixel values less than $x_*^{(k)} - r$ are shifted accordingly such that

$$m^{(k+1)}(x) = \begin{cases} m^{(k)}(x) & x < x_- \\ m^{(k)}\left(x - \sum_{l=x_-}^{x} \mathbb{1}(g^{(k)}(l) = 0)\right) & x_- \leq x \leq x_*^{(k)} - r \\ x_*^{(k)} & x_*^{(k)} - r < x < x_*^{(k)} \\ m^{(k)}(x) & x > x_*^{(k)} \end{cases} \tag{3.10}$$

where x_- is the location of the r^{th} zero in $g^{(k)}$ counting down from $x_*^{(k)}$.

After the estimate of the contrast enhancement mapping is updated, the set X is updated using the equation

$$X^{(k+1)} = \{x | m^{(k+1)}(x) \in (X^{(k)} \cup x_*^{(k)})\}. \tag{3.11}$$

For all pixel values not in the set $X^{(k+1)}$, the estimate of the unaltered pixel value histogram is updated by

$$g^{(k+1)}(x) = \sum_l g^{(0)}(l)\, \mathbb{1}(m^{(k+1)}(l) = x). \tag{3.12}$$

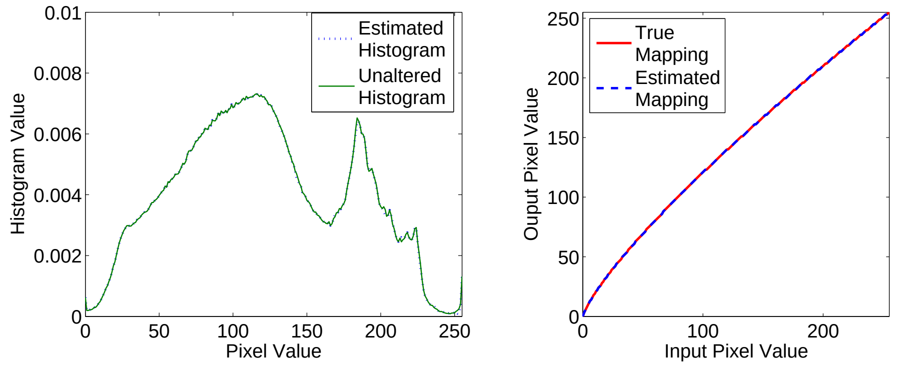


Figure 3.6: Left: Unaltered pixel value histogram and its estimate. Right: Contrast enhancement mapping and its estimate.

The value of $g^{(k+1)}$ is then interpolated at pixel values in the set $X^{(k+1)}$, and appropriately normalized such that

$$g^{(k+1)}(x) = \hat{g}^{(k+1)}(x)\frac{g^{(0)}(m^{(k+1)}(x))}{\sum_{t|m^{(k+1)}(t)=m^{(k+1)}(x)} \hat{g}^{(k+1)}(t)}. \tag{3.13}$$

The iteration is terminated when either the maximum value of $\left|\frac{\hat{g}^{(k)}(x)-g^{(k)}(x)}{\hat{g}^{(k)}(x)}\right|$ falls below a preset threshold or when no zeros remain in $g^{(k)}$.

3.4 Results

To test the performance of our proposed algorithm, we applied it to the contrast enhanced pixel value histogram shown in Fig. 3.4. The resulting estimate of the unaltered pixel value histogram is shown in Fig. 3.6 along with the true unaltered pixel value histogram. When we compare the estimated histogram to the true one, we find very few differences between the two. Estimation errors occur primarily in regions where the unaltered histogram's first difference changes values abruptly. The rightmost image in Fig. 3.6 shows the estimate of the contrast enhancement mapping used to modify the image. This is plotted along with the true contrast enhancement mapping $y = 255(\frac{x}{255})^{\gamma}$, which corresponds to the gamma correction with $\gamma = 0.8$. Our algorithm was able to perfectly estimate this contrast enhancement mapping.

To demonstrate our algorithm's ability to operate on an image modified by a nonstandard form of contrast enhancement, we used it to obtain estimates of the unaltered histogram and contrast enhancement mapping from the pixel value histogram displayed at the bottom right of Fig. 3.7. The contrast enhancement mapping used to modify the image is shown at the right of Fig. 3.7 along with our estimate of the mapping. The top left image in Fig. 3.7 shows the pixel value histogram of the image before contrast enhancement as well as our estimate of the unaltered histogram. These results indicate that our algorithm is capable of achieving accurate mapping and histogram estimates from images modified by a large class of contrast enhancement operations not considered in [8].

3.5 Summary

In this chapter, we proposed an iterative algorithm to jointly estimate an image's unaltered pixel value histogram as well as the contrast enhancement mapping used to modify the image given only a contrast enhanced version of the image. We used a probabilistic model of an image's histogram to identify the histogram entries most likely to correspond to contrast enhancement artifacts. We then used this model along with knowledge of how contrast enhancement modifies an image's

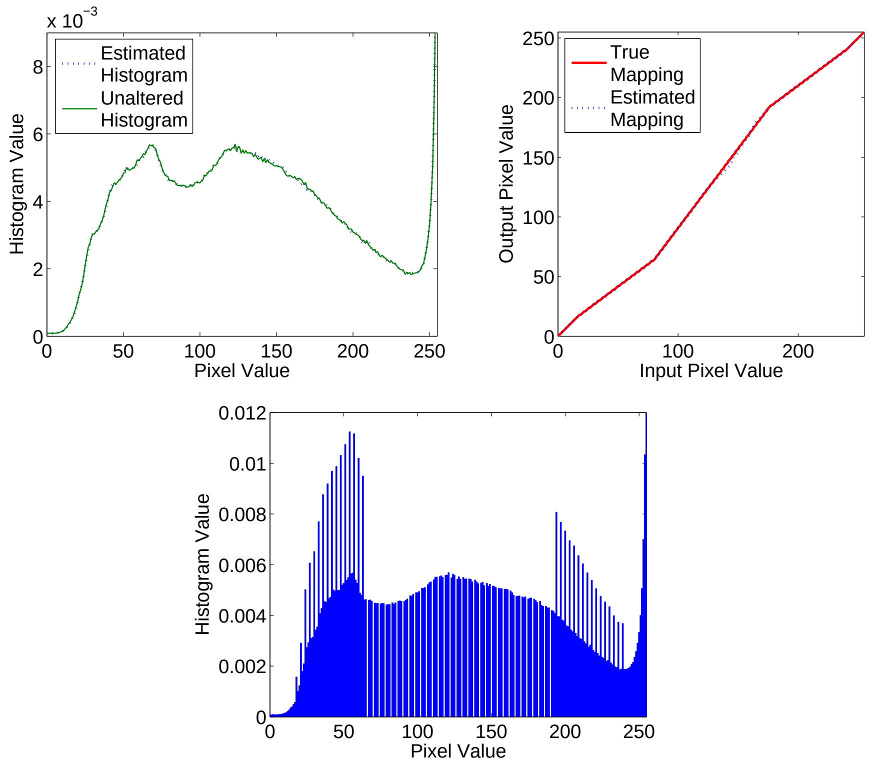


Figure 3.7: Top Left: Unaltered pixel value histogram and its estimate. Top Right: Contrast enhancement mapping and its estimate. Bottom: Pixel value histogram after contrast enhancement.

histogram to obtain our unaltered histogram and contrast enhancement mapping estimates. Simulation results indicate that our algorithm is capable of providing accurate estimates even when nonstandard forms of contrast enhancement are applied to an image.

Chapter 4

Anti-Forensics of Digital Image Compression

Due to the widespread availability of digital cameras and the rise of the Internet as a means of communication, digital images have become an important method of conveying visual information. Unfortunately, the ease with which digital images can be manipulated by photo-editing software has created an environment where the authenticity of digital images is often in doubt. To prevent digital image forgeries from being passed off as unaltered originals, researchers have developed a variety of digital image forensic techniques. These techniques are designed to determine an image's originating camera [5], trace its processing history [28], and determine its authenticity [1, 52], all without relying on an extrinsically inserted watermark or access to the original image. Instead, these techniques make use of *intrinsic fingerprints* introduced into an image by editing operations or the image formation process itself [44].

Image compression fingerprints are of particular forensic significance due the fact that most digital images are subjected to compression either by the camera used to capture them, during image storage, or for the purposes of digital transmission over the Internet. Techniques have been developed to determine if an image saved in a lossless format has ever undergone JPEG compression [7, 26] or other types of image compression including wavelet-based techniques [26]. If evidence of

JPEG compression is detected, the quantization table used during compression can be estimated [7]. Because most digital cameras and image editing software use proprietary JPEG quantization tables when compressing an image, an image's origin can be identified by matching the quantization tables used to compress the image with those in a database of quantization table and camera or software pairings [9]. If the quantization tables are matched with those used by image editing software, the authenticity of the image can be called into question. Recompressing an image which has previously been JPEG compressed, also known as double JPEG compression, can be detected [32, 33] and the quantization table used during the initial application of JPEG compression can be estimated. Localized evidence of double JPEG compression can be used to identify image forgeries [16] as well as localized mismatches in an image's JPEG block artifact grid [57].

Though many existing forensic techniques are capable of detecting a variety of standard image manipulations, they do not account for the possibility that *anti-forensic* operations may be designed and used to hide image manipulation fingerprints. This is particularly important because it calls into question the validity of forensic results indicating the absence of image tampering. It may be possible for an image forger familiar with signal processing to secretly develop anti-forensic operations and use them to create undetectable image forgeries. As a result, several existing forensic techniques may contain unknown vulnerabilities.

In order to combat the creation and spread of undetectable image forgeries, it is necessary for image forensics researchers themselves to develop and study anti-forensic operations. By doing so, researchers can be made aware of which forensic techniques are capable of being deceived, thus preventing altered images from being represented as authentic and allowing forensic examiners to establish a degree of confidence in their findings. Furthermore, it is likely that many anti-forensic operations will leave behind detectable fingerprints of their own. If these fingerprints can be discovered, forensic techniques can be designed to detect the use of anti-forensic operations. It is also possible that anti-forensic operations may be used to provide intellectual property protection. This would be done by integrating them into digital image and video cameras to prevent the reverse engineering of proprietary signal processing components through digital forensic means.

At present, very little anti-forensics research has been published. To the best of our knowledge, the only prior work studying digital image anti-forensics are techniques to remove traces of image resizing and rotation [21], to forge the photo-response non-uniformity noise fingerprint left in an image by a digital camera's electronic sensor [12], and to artificially synthesize color filter array artifacts [22]. In this chapter, we present a set of anti-forensic operations capable of removing compression fingerprints from digital images [48]. Since most modern lossy image compression techniques involve transform coding, we propose a framework for the removal of quantization fingerprints from a compressed image's transform coefficients by adding *anti-forensic dither* to them. We use this framework to develop anti-forensic operations to remove quantization artifacts from the DCT coefficients of JPEG compressed images and from the wavelet coefficients of wavelet based schemes such as JPEG 2000, SPIHT, and EZW [46, 49]. Additionally, we propose an anti-forensic operation to remove statistical traces of blocking artifacts from

JPEG compressed images. We then experimentally demonstrate that our proposed anti-forensic operations can be used to fool a variety of compression fingerprint based forensic algorithms designed to detect single and double JPEG compression, wavelet-based image compression, determine an image's origin, and detect cut-and-paste image forgeries [50].

The organization of this chapter is as follows. In Section 4.1, we discuss the quantization fingerprints left by image transform coders and propose a generalized framework for their removal. We adapt this framework for use with JPEG compression in Section 4.2 and wavelet-based compression in Section 4.3. In Section 4.4, we propose an anti-forensic technique capable of removing statistical traces of blocking artifacts. We present the results of several experiments designed to evaluate the performance of each of our proposed anti-forensic techniques in Section 4.5. In Section 4.6 we discuss how these techniques can be used to render certain forms of image tampering such as double JPEG compression, cut-and-paste image forgery, and image origin falsification undetectable through compression history based forensic means. Finally, we summarize this chapter in Section 4.7.

4.1 Anti-Forensic Framework

Virtually all modern lossy image compression techniques are subband coders, which are themselves a subset of transform coders. Transform coders operate by applying a mathematical transform to a signal, then compressing the transform coefficients. Subband coders are transform coders that decompose the signal into different frequency bands or subbands of transform coefficients. Typical lossy image compression techniques operate by applying a two-dimensional invertible transform, such as the discrete cosine transform (DCT) or discrete wavelet transform (DWT), to an image as a whole, or to each set of pixels within an image that has been segmented into a series of disjoint sets. As a result, the image or set of pixels is mapped into multiple subbands of transform coefficients, where each transform coefficient is denoted $X \in \mathbb{R}$.

Once obtained, each transform coefficient must be mapped to a binary value both for storage and to achieve lossy compression. This is achieved through the process of quantization, in which the binary representation $\hat{X}$ of the transform coefficient X is assigned the value $\hat{x}$ according to the equation

$$\hat{X} = \hat{x} \quad \text{if } b_k \leq X < b_{k+1}, \tag{4.1}$$

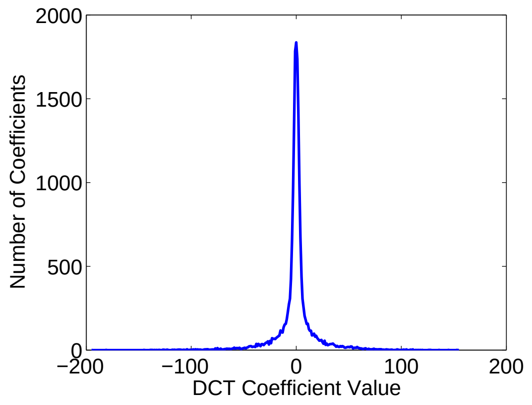
where b_k and b_{k+1} denote the boundaries of the quantization interval over which X maps to the value $\hat{x}$. Because some subbands of transform coefficients are less perceptually important than others, thus can accomodate greater loss during the quantization process, the set of quantization interval boundaries is chosen differently for each subband. After each transform coefficient is given a binary representation, the binary values are reordered into a single bit stream which is often subjected to lossless compression.

When the image is decompressed, the binary bit stream is first rearranged into its original two dimensional form. Each decompressed transform coefficient Y is assigned a value through dequantization. During this process each binary value is mapped to a quantized transform coefficient value q belonging to the discrete set $\mathcal{Q} = \{\dots, q_{-1}, q_0, q_1, \dots\}$. Each dequantized transform coefficient value can be directly related to its corresponding original transform coefficient value by the equation

$$Y = q_k \quad \text{if } b_k \leq X < b_{k+1}. \tag{4.2}$$

After dequantization, the inverse transform is performed on the set of transform coefficients and the resulting values are projected back into the set of allowable pixel values $\mathcal{P} = \{0, \dots, 255\}$. If the image was segmented, this process is repeated for each segment and the decompressed segments are joined together to create the decompressed image; otherwise, this process reconstructs the decompressed image as a whole.

By performing image compression in this manner, a distinct fingerprint is introduced into the transform coefficients of an image. When examining an unaltered image's transform coefficient values within a particular subband, they will likely be distributed according to a smooth, continuous distribution. This is not the case for images which have undergone image compression, since the processes of quantization and dequantization force the transform coefficients of a compressed image to take values within the discrete set $\mathcal{Q}$. In practice, the act of projecting the decompressed pixel values perturbs the transform coefficient values, though the transform coefficients of a previously compressed image still cluster tightly around elements of $\mathcal{Q}$. These fingerprints, known as transform coefficient quantization artifacts, are used by the majority of compression artifact based forensic techniques to identify



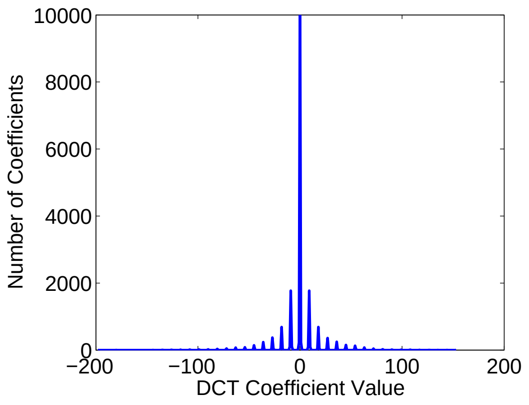


Figure 4.1: Left: Histogram of DCT coefficients from an uncompressed image. Right: Histogram of DCT coefficients from the same image after JPEG compression

single or double compression, determine an image's origin, or identify image forgeries. They can be clearly seen in Figure 4.1, which shows the histogram of one subband of DCT coefficients from an image before and after JPEG compression.

If the image was divided into segments during compression, another compression fingerprint may arise. Because of the lossy nature of image transform coding, pixel domain discontinuities often arise across the boundaries of these segments. Research has shown that these discontinuities can be statistically detected even when they are not visible [7]. These discontinuities are known as blocking artifacts, since in the majority of cases the image segments take the form of square blocks. While important, these fingerprints are less frequently used by forensic algorithms, and their anti-forensic removal will be discussed in Section 4.4.

To remove transform coefficient quantization artifacts from a compressed image we propose the following generalized framework. First, we model the distribution of the transform coefficients for a given subband prior to quantization using a parameteric model $P(X = x) = f(x, \theta)$ with parameter θ. Next, we estimate the value of θ from the quantized transform coefficients. We then anti-forensically modify each quantized transform coefficient by adding specially designed noise, which we refer to as *anti-forensic dither*, to its value according to the equation

$$Z = Y + D, \tag{4.3}$$

where D is the anti-forensic dither and Z is the anti-forensically modified coefficient. The distribution of the anti-forensic dither is chosen so that it corresponds to a renormalized and recentered segment of the model distribution for that subband, where the segment is centered at the quantized coefficient value and the segment's length is equal to the length of the quantization interval. Because the probability that the quantized coefficient value is q_k is given by

$$P(Y = q_k) = \int_{b_k}^{b_{k+1}} f(x, \theta) dx, \tag{4.4}$$

the anti-forensic dither's distribution is given by the formula

$$P(D = d | Y = q_k) = \frac{f(q_k + d, \theta)}{\int_{b_k}^{b_{k+1}} f(x, \theta) dx} \mathbb{1}(b_k \leq q_k + d < b_{k+1}). \tag{4.5}$$

As a consequence of this, the anti-forensic dither distribution will be conditionally dependent not only upon the value of θ, but on the value of the coefficient to which the dither is to be added as well.

Choosing the anti-forensic dither distribution in this manner yields two main benefits; the anti-forensically modified coefficient distribution will theoretically match the transform coefficient distribution before quantization and an upper bound can be placed on the distance between each unquantized transform coefficient and its anti-forensically modified counterpart. To prove the first property, we make use of the fact that $P(Z = z|Y = q_k) = P(D = z - q_k|Y = q_k)$. We then use the law of total probability to write an expression for the anti-forensically modified coefficient distribution as follows:

$$\begin{aligned} P(Z = z) &= \sum_k P(Z = z|Y = q_k)P(Y = q_k), \\ &= \sum_k \left(\frac{f(q_k + (z - q_k), \theta)}{\int_{b_k}^{b_{k+1}} f(x, \theta)dx} \mathbb{1}(b_k \le q_k + d < b_{k+1}) \right) \int_{b_k}^{b_{k+1}} f(x, \theta)dx, \\ &= \sum_k f(z, \theta) \mathbb{1}(b_k \le q_k + d < b_{k+1}), \\ &= f(z, \theta), \end{aligned} \tag{4.6}$$

thus proving $P(Z = z) = P(X = z)$. This is important because it proves that forensic analysis of the transform coefficient distribution of an image should be unable to distinguish an unaltered image from an anti-forensically modified one, provided that the distribution of unmodified coefficients is modeled accurately and the parameter θ is correctly estimated.

An upper bound can be placed on the distance between an unquantized transform coefficient value and its anti-forensically modified counterpart by first examining the distance between an unquantized coefficient and its corresponding quantized value. Assuming that each quantized value lies at the center of its corresponding quantization interval, this distance can be trivially bounded as follows

$$|X - Y| \le \max_k \tfrac{1}{2}|b_k - b_{k+1}|. \tag{4.7}$$

Because each anti-forensically modified coefficient value must lie within the quantization interval encompassing the modified quantized coefficient value, the bound placed on $|X-Y|$ also holds for $|Y-Z|$. As a result, the distance between an unquantized and anti-forensically modified transform coefficient value is upper bounded by

$$|X - Z| \leq \max_k |b_k - b_{k+1}|. \tag{4.8}$$

If the transform coefficients are subjected to uniform quantization, i.e. $|b_k - b_{k+1}| = Q$ for all k, this bound can be rewritten as $|X - Z| \leq Q$. Though it is often difficult to analytically translate distortion introduced in the transform coefficient domain to the pixel domain, this upper bound demonstrates that the amount of distortion introduced into the image through anti-forensic modification is determined by the compression strength.

4.2 JPEG Anti-Forensics

In this section, we provide a brief overview of JPEG compression, then present our anti-forensic technique designed to remove compression fingerprints from a JPEG compressed image's DCT coefficients.

4.2.1 JPEG Compression Overview

For a grayscale image, JPEG compression begins by segmenting an image into a series of nonoverlapping 8×8 pixel blocks, then computing the two-dimensional DCT of each block. The resulting transform coefficients are then quantized by dividing each coefficient value by its corresponding entry in predetermined quantization matrix Q, then rounding the resulting value to the nearest integer. Accordingly, a quantized DCT coefficient at the (i, j) block position is represented by the value $\hat{X} = \text{round}(\frac{X}{Q_{i,j}})$. Finally, the binary representations of each quantized DCT coefficient are reordered into a single bit stream using the zigzag scan order, then losslessly encoded. Color images are compressed in a similar manner, however they require additional preprocessing. First, the image is transformed from the RGB to the YCbCr color space. Next, the chrominance layers are typically downsampled by a factor of two in both the horizontal and vertical directions. After this has been performed, compression continues as if each color layer were an independent grayscale image.

A JPEG image is decompressed by first losslessy decoding the bit stream, then rearranging the integer representations of the quantized DCT coefficients back into their original 8×8 block form. Next, the DCT coefficient values are dequantized by multiplying the integer representation of each DCT coefficient value by its corresponding entry in the quantization matrix. The inverse DCT of each block of coefficients is computed and the resulting pixel values are projected back into the set $\mathcal{P}$ of allowable pixel values. The decompressed grayscale image or color layer is then reassembled from the series decoded blocks. If a color image that was subject to chrominance layer downsampling is decompressed, each of the downsampled layers are returned to their original size through interpolation, then the image is transformed back into the RGB color space.

As was discussed in Section 4.1, JPEG compression will result in two forensically significant fingerprints: DCT coefficient quantization fingerprints and blocking artifacts. DCT coefficient quantization fingerprints, which can be seen in the DCT coefficient histograms displayed in Figure 4.1, correspond to the clustering of DCT coefficient values around integer multiples of their corresponding entry in the quantization. This occurs because a quantized DCT coefficient value Y is related to its unquantized counterpart by the equation $Y = Q_{i,j}\,\text{round}(\frac{X}{Q_{i,j}})$. JPEG blocking artifacts are the discontinuities which occur across the 8×8 pixel block boundaries that arise due to pixel value perturbations caused by DCT coefficient quantization. Anti-forensic removal of these fingerprints will be discussed in detail in Section 4.4.

4.2.2 DCT Coefficient Quantization Fingerprint Removal

In accordance with the anti-forensic framework which we outlined in Section 4.1, we begin by modeling the distribution of coefficient values within a particular AC DCT subband using the Laplace distribution [23]

$$P(X = x) = \tfrac{\lambda}{2} e^{-\lambda|x|}. \tag{4.9}$$

Though we use this model for each AC subband of the DCT, each subband will have its own unique value of λ. Using this model and the quantization rule described above, the coefficient values of an AC subband of DCT coefficients within a previously JPEG compressed image will be distributed according to the discrete Laplace distribution

$$P(Y = y) = \begin{cases} 1 - e^{-\lambda Q_{i,j}/2} & \text{if } y = 0, \\ e^{-\lambda|y|} \sinh(\frac{\lambda Q_{i,j}}{2}) & \text{if } y = kQ_{i,j}, \\ 0 & \text{otherwise}, \end{cases} \tag{4.10}$$

where $k \in \mathbb{Z}$, $k \neq 0$.

To anti-forensically modify a previously JPEG compressed image, we first perform the initial steps in JPEG compression (i.e. color space transformation, segmentation into blocks, DCT) to obtain a set of DCT coefficients from the image. Because the final stages of JPEG decompression involve projecting the decompressed pixel values back into $\mathcal{P}$, the DCT coefficient values obtained from the image will be perturbed from their quantized values. We assume these perturbations are small enough that they do not move a coefficient value into a different quantization interval. As a result, the quantized coefficient values can be obtained by repeating the quantization process upon the perturbed coefficients Y' so that $Y = Q_{i,j}\,\mathrm{round}(\frac{Y'}{Q_{i,j}})$.

Next, we obtain a maximum likelihood estimate the model parameter λ independently for each AC subband of DCT coefficients using the quantized coefficients [36]. By doing this, we can use our model obtain an estimate of each AC subband's coefficient distribution before JPEG compression. We define $N = N_0 + N_1$ as the total number of observations of the current DCT subband, N_0 as the number DCT subband of coefficients taking the value zero, N_1 as the number of non-zero valued coefficients, and $S = \sum_{k=1}^{N} |y_k|$. The model parameter estimate, which we denote $\hat{\lambda}$, is calculated using the equation

$$\hat{\lambda} = -\frac{2}{Q_{i,j}} \ln(\gamma), \tag{4.11}$$

where γ is defined as

$$\gamma = \frac{-N_0 Q_{i,j}}{2NQ_{i,j} + 4S} + \frac{\sqrt{N_0^2 Q_{i,j}^2 - (2N_1 Q_{i,j} - 4S)(2NQ_{i,j} + 4S)}}{2NQ + 4S}. \tag{4.12}$$

After λ has been estimated, we add anti-forensic dither to each DCT coefficient in an AC subband. Because we model the coefficient distribution before quantization using the Laplace distribution, the expression for the anti-forensic dither's distribution given in (4.5) simplifies to one of two equations depending upon the magnitude of the quantized DCT coefficient value to which the dither is added. For zero-valued quantized coefficients, the anti-forensic dither distribution is chosen to be

$$P(D = d|Y = 0) = \begin{cases} \frac{1}{c_0} e^{-\hat{\lambda}|d|} & \text{if } \frac{-Q_{i,j}}{2} \geq n > \frac{Q_{i,j}}{2}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.13}$$

where $c_0 = 1 - e^{-\hat{\lambda} Q_{i,j}/2}$. The distribution of the anti-forensic dither added to nonzero quantized DCT coefficients is

$$P(D = d|Y = q_k) = \begin{cases} \frac{1}{c_1} e^{-\operatorname{sgn}(q_k)\hat{\lambda}(d + Q_{i,j}/2)} & \text{if } \frac{-Q_{i,j}}{2} \geq n > \frac{Q_{i,j}}{2}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.14}$$

where $c_1 = \frac{1}{\hat{\lambda}}(1 - e^{-\hat{\lambda} Q_{i,j}})$. An important benefit of the anti-forensic dither distributions taking these forms is that they reduce the complexity of generating the anti-forensic dither. Rather than drawing dither samples from a number of distributions equal to the number of distinct quantized DCT coefficient values within an AC subband, anti-forensic dither samples need only to be drawn from one of the two distributions displayed in (4.13) and (4.14).

As we demonstrated for the general case in (4.6), using these anti-forensic dither distributions will ensure that the distribution of anti-forensically modified DCT coefficients within an AC subband will match its modeled unquantized coefficient distribution. By using the expressions for the quantized coefficient distribution as well as the anti-forensic dither distribution given in (4.10), (4.13), and (4.14), and using the law of total probability we may write

$$\begin{aligned}P(Z=z) &= \sum_k P(Z=z|Y=q_k)P(Y=q_k) \\ &= \sum_{q_k \neq 0} \tfrac{1}{c_1} e^{-\operatorname{sgn}(q_k)\hat{\lambda}(z-q_k+Q_{i,j}/2)} e^{-\hat{\lambda}|q_k|} \sinh(\tfrac{\hat{\lambda}Q_{i,j}}{2})\, \mathbb{1}(|z-q_k| \leq \tfrac{Q_{i,j}}{2}) \\ &\quad + \tfrac{1}{c_0} e^{-\hat{\lambda}|z|}(1-e^{-\hat{\lambda}Q_{i,j}/2})\, \mathbb{1}(|z| \leq \tfrac{Q_{i,j}}{2}) \\ &= \tfrac{\hat{\lambda}}{2} e^{-\hat{\lambda}|z|}.\end{aligned} \tag{4.15}$$

In most quantization tables, larger quantization step sizes are used for high frequency DCT subbands because changes to these subbands are less perceptually significant. Furthermore, it has been observed that the variance of coefficient values within a DCT subband decreases as one moves from low frequency to high frequency subbands. Because of this, all of the coefficient values of certain high frequency DCT subbands will be quantized to zero in some images during JPEG compression. Correspondingly, no estimate of λ can be obtained for these DCT subbands, rendering us unable to anti-forensically modify their coefficient values. Fortunately, the DCT coefficient value perturbations caused by the final steps in JPEG decompression result the coefficient values of these subbands taking on a plausible distribution, as can be seen in Figure 4.2. As a result, we do not need to anti-forensically modify the coefficients of these DCT subbands.

Because the distribution of the DC subband of DCT coefficients varies greatly from image to image, no accurate parametric model for this distribution exists. Instead, we model the distribution of DC subband of unquantized DCT coefficients as being uniformly distributed within a quantization interval. As a consequence, we are able to create a set of anti-forensically modified coefficients whose distribution

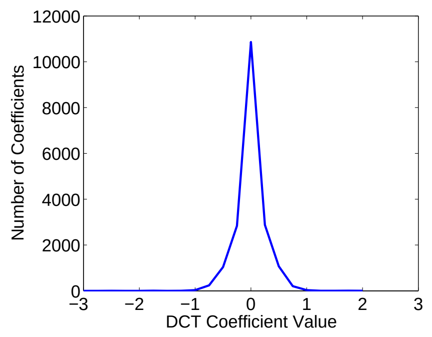


Figure 4.2: Histogram of perturbed DCT coefficient values from a DCT subband in which all coefficients were quantized to zero during JPEG compression.

approximates the unquantized distribution by adding uniformly distributed anti-forensic dither to the quantized DC subband of DCT coefficients. The dither is chosen to be zero mean over a support interval equal in length to the quantization interval so that

$$P(D = d) = \begin{cases} \frac{1}{Q_{i,j}} & \text{if } \frac{-Q_{i,j}}{2} \leq n < \frac{Q_{i,j}}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.16}$$

Though this could in theory introduce step discontinuities into the distribution of the DC subband of anti-forensically modified DCT coefficients, we have experimentally observed that this is rarely the case. The absence of step discontinuities from the empirical distribution of anti-forensically modified coefficients is likely due to the fact that the dynamic range of DC DCT values is typically sufficiently large in comparison to the quantization interval that relatively few DC coefficients are quantized to any given value. As a result, too few anti-forensically modified coefficient values exist over an interval for step discontinuities to be discernible.

After the anti-forensically modified DCT coefficients are obtained, the inverse DCT of each block of coefficients is performed and the resulting blocks of pixel values

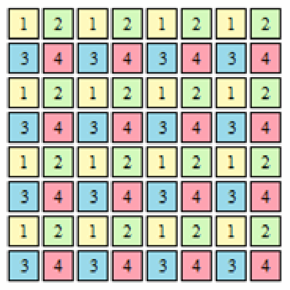


Figure 4.3: Chrominance layer reconstruction interleaving pattern.

are assembled into the anti-forensically modified image. If a color image subjected to chrominance layer downsampling during JPEG compression undergoes anti-forensic modification, a number equal to the downsampling factor of independently generated anti-forensically modified versions of each downsampled chrominance layer is created. Each independent version of the anti-forensically modified downsampled chrominance layer is then interleaved to create one equal in size to the full sized image. For images that undergo chrominance layer downsampling by a factor of two in each direction as is most commonly the case, the anti-forensically modified downsampled layers are interleaved using the pattern shown in Figure 4.3.

4.3 Wavelet-Based Image Compression Anti-Forensics

In this section, we begin by providing a brief overview of several wavelet-based image compression techniques and their forensically significant compression fingerprints. After this, we present our anti-forensic technique designed to remove compression fingerprints from the wavelet coefficients of an image compressed using a wavelet-based technique.

4.3.1 Wavelet-Based Compression Overview

Though several wavelet-based image compression techniques exist such as SPIHT, EZW, and most popularly JPEG 2000, they all operate in a similar fashion and leave behind similar compression fingerprints. Techniques such as JPEG 2000 begin compression by first segmenting an image into fixed sized nonoverlapping rectangular blocks known as 'tiles', while others operate on the image as a whole. Next, the two-dimensional discrete wavelet transform (DWT) of the image or each image tile is computed, resulting in four subbands of wavelet coefficients. Because these subbands correspond to either high (H) or low (L) frequency DWT coefficients in each spatial dimension, the four subbands are referred to using the notation LL, LH, HL, and HH. The DWT of the LL subband is computed an additional $M-1$ times, resulting in an M-level wavelet decomposition of the image or tile.

After this, tree-based compression techniques such as SPIHT or EZW divide the set of DWT into separate bit planes which are each processed independently. Within each bit plane, a tree-like structure known as a significance map is constructed detailing the locations of nonzero coefficients at that bit level [37]. Because the locations of zero-valued bit plane coefficients are correlated across DWT subbands, this allows for the efficient storage of each bit plane. The significance maps of each bit plane are then reordered into a single bit stream, with the map of the most significant bit plane occurring at the beginning of the bit stream, then proceeding in descending order of significance. To achieve lossy compression, the bit stream is truncated to a fixed number of bits according to a predefined bit budget.

JPEG 2000 achieves lossy compression in an alternate manner [39]. First, each subband of DWT coefficients is independently quantized using their own fixed quantization step size. Next, the binary representations of the quantized coefficients are divided into bit planes and separated into code blocks which are then entropy coded and reordered into a single bit stream. Because compression is achieved through quantization, the bit stream does not undergo truncation.

Image decompression begins by obtaining the set of DWT coefficients from the bit stream. Tree-based techniques such as SPIHT or EZW accomplish this by reforming the set of significance maps from the bit stream, then using them to recreate each bit plane. During this process, bit plane data which was truncated during compression is replaced with zeros. For images compressed using JPEG 2000, the integer representation of each coefficient is decoded from the bit stream and the quantized DWT coefficient values are obtained through the dequantization process. Finally, the inverse DWT of the image or image tile is computed and resulting pixel values are projected back into the set $\mathcal{P}$ of allowable pixel values. If tiling was used, the full image is reassembled from the set of reconstructed tiles.

While these image compression techniques achieve lossy compression through different processes, they each introduce DWT coefficient quantization fingerprints into an image. For JPEG 2000, this is fairly obvious as the quantization and dequantization process cause the DWT coefficients in decompressed images to cluster around integer multiples of their respective subband's quantization step size. In SPIHT and related algorithms, a similar process occurs because bit stream truncation results in the loss of the bottom several bit planes. As a result, only the

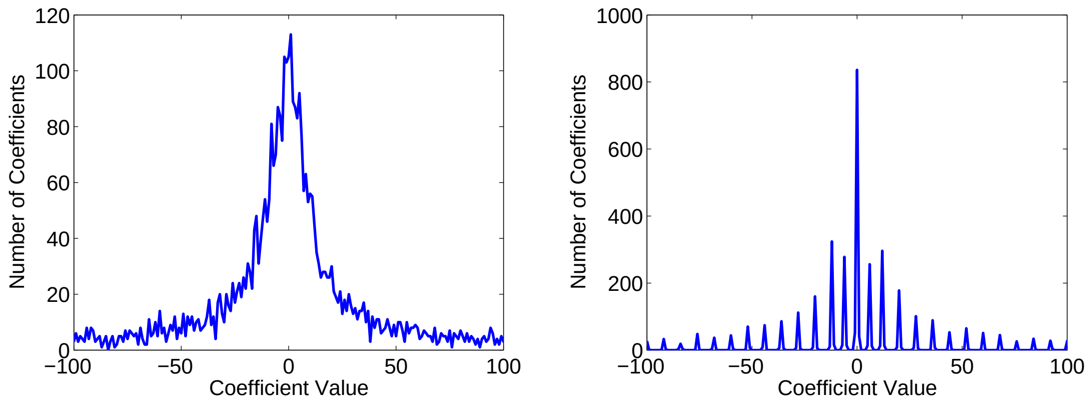


Figure 4.4: Left: Histogram of wavelet coefficients from an uncompressed image. Right: Histogram of wavelet coefficients from the same image after SPIHT compression.

n most significant bits of each DWT coefficient are retained. This is equivalent to applying the quantization rule in (4.2) where X is a DWT coefficient from an uncompressed image, Y is the corresponding DWT coefficient in its SPIHT compressed counterpart, and

$$q_k = \begin{cases} b_k & \text{if } k \geq 1, \\ 0 & \text{if } k = 0, \\ b_{k+1} & \text{if } k \leq -1. \end{cases} \tag{4.17}$$

These DWT coefficient quantization fingerprints can be observed when viewing the distribution of coefficient values within a particular DWT subband as seen in Figure 4.4. Additionally, if the image was tiled during compression, tiling artifacts similar to JPEG blocking artifacts may occur in an image.

4.3.2 DWT Coefficient Quantization Fingerprint Removal

We model the distribution of coefficient values within a DWT subband of an uncompressed image using the Laplace distribution [24]

$$P(X = x) = \tfrac{\lambda}{2} e^{-\lambda|x|}. \tag{4.18}$$

Because the manner in which JPEG 2000 employs DWT coefficient quantization is identical to the way in which JPEG performs DCT coefficient quantization, the distribution of coefficients within a particular DWT subband in a previously JPEG 2000 compressed image is given by (4.10), where $Q_{i,j}$ is replaced by the quantization step size used for that DWT subband. As a result, the DWT coefficients in a previously JPEG 2000 compressed image can be anti-forensically modified using the method outlined in Section 4.2.2. The remainder of this section will focus primarily on SPIHT and other tree-based compression schemes whose DWT coefficient quantization rules are given by (4.2) and (4.17). Using these equations along with (4.18), the distribution of coefficient values within a DWT subband in an image previously SPIHT or similarly compressed is given by

$$P(Y = q_k) = \begin{cases} \tfrac{1}{2}(e^{-\lambda b_k} - e^{-\lambda b_{k+1}}) & \text{if } k \geq 1, \\ 1 - \tfrac{1}{2}(e^{\lambda b_0} + e^{-\lambda b_1}) & \text{if } k = 0, \\ \tfrac{1}{2}(e^{\lambda b_{k+1}} - e^{\lambda b_k}) & \text{if } k \leq -1. \end{cases} \tag{4.19}$$

In order to anti-forensically modify an image previously compressed using a wavelet-based technique, we must first obtain the set of quantized DWT coefficients from the compressed image. To do this, we repeat the first several steps of compression including tiling the image if necessary and computing the DWT of the image or set of image tiles. Since the process of projecting the decompressed pixel values back into $\mathcal{P}$ during decompression perturbs the DWT coefficient values, the quantized coefficient values Y must be recovered from the perturbed coefficient values Y'. This can be done for previously JPEG 2000 compressed images by simply reapplying the quantization rule used during compression. For images compressed using SPIHT and related techniques, this is not appropriate since DWT coefficients are quantized to values on the edge of each quantization interval and the perturbations can move these values into a different quantization interval. Instead, we assume that the perturbations are sufficiently small and recover the quantized coefficient values according to the rule $Y = q_k$ if $\frac{q_k+q_{k-1}}{2} \leq Y' < \frac{q_{k+1}+q_k}{2}$.

Once the quantized DWT coefficient values have been recovered, we estimate the parameter λ for each DWT subband. This is done by fitting the nonzero entries of the histogram of each DWT subband's coefficient values to the function

$$h_k = ce^{-\hat{\lambda}|q_k|}, \tag{4.20}$$

where $\hat{\lambda}$ is the estimated value of λ, h_k denotes the the histogram value at q_k, and c is a scaling constant. By linearizing (4.20) by taking the logarithm of both sides of the equation, this fitting problem can be reformulated as the least squares minimization problem

$$\min_{\hat{\lambda},c} \sum_k h_k(\log h_k - \log c + \hat{\lambda}|q_k|)^2, \tag{4.21}$$

where the model errors have been weighted by h_k, the number of observations of each quantized DWT coefficient value. To solve this minimization problem, we take the derivative with respect to $\hat{\lambda}$ and c of the function to be minimized in (4.21), set these derivatives to zero, then reformulate the resulting equations into the matrix

$$\begin{bmatrix} \sum_k h_k & \sum_k |q_k| h_k \\ \sum_k |q_k| h_k & \sum_k |q_k|^2 h_k \log h_k \end{bmatrix} \begin{bmatrix} \log c \\ -\hat{\lambda} \end{bmatrix} = \begin{bmatrix} \sum_k h_k \log h_k \\ \sum_k |q_k| h_k \log h_k \end{bmatrix}, \tag{4.22}$$

We then solve (4.22) for $\hat{\lambda}$ and c.

Though this estimate yields satisfactory results under ideal circumstances, in practice bit stream truncation effects often lead to a mismatch between our model of the DWT coefficient distribution and the histogram of DWT coefficient values obtained from a previously compressed image. Because the point at which the bit stream is truncated rarely corresponds to the boundary between bit planes, a significant number of entries in the lowest bit plane are often set to zero. This results in an artificial decrease in the number of DWT coefficients taking the values q_1 and q_{-1}, and an artificial increase in the number of coefficients taking the value 0 over the number of coefficients predicted by our model. This, in turn, leads to an estimation bias which we compensate for using an iterative process to refine our estimate of λ.

We initialize our iterative procedure by setting $h_k^{(1)} = h_k$ for all k, $\hat{\lambda}^{(0)} = 0$, and the initial iteration index to $i = 1$. We then repeat the following steps until the termination criteria is met:

1. Estimate $\hat{\lambda}^{(i)}$ and $c^{(i)}$ by solving (4.22) using the current histogram iterate $\hat{h}^{(i)}$ in lieu of h.

2. Update the histogram estimate according to the equation:

$$\hat{h}_k^{(i+1)} = \begin{cases} c^{(i)} & \text{if } k = 0, \\ h_k + \frac{1}{2}(h_0 - c^{(i)}) & \text{if } k = \pm 1, \\ h_k & \text{otherwise.} \end{cases} \tag{4.23}$$

3. Terminate if $\frac{\hat{\lambda}^{(i)} - \hat{\lambda}^{(i-1)}}{\hat{\lambda}^{(i)}} < \tau$, where τ is a user defined threshold. Otherwise, set $i = i + 1$ and return to Step 1.

After this process is terminated, the final value of $\hat{\lambda}^{(i)}$ is retained as the parameter estimate $\hat{\lambda}$.

Before anti-forensic dither can be added to the quantized DWT coefficients, the mismatch between between our model of the DWT coefficient distribution and the true DWT coefficient histogram must be corrected. Because bit stream truncation can occur anywhere within the least significant retained bit plane, we cannot accurately predict the number of components of that bit plane that will be set to zero. Accordingly, we cannot appropriately adjust our model to take partial bit plane truncation into account. Instead, we modify the DWT coefficient histogram to match our model by changing a number of DWT coefficient values from 0 to q_1 or q_{-1}. We calculate N_e, the number of zero valued DWT coefficients in excess of what our model predicts using the equation

$$N_e = h_0 - N_s(1 - \tfrac{1}{2}(e^{\hat{\lambda} b_0} + e^{-\hat{\lambda} b_1})), \tag{4.24}$$

where N_s is the total number of DWT coefficients in the current subband. We then randomly change the values of $\frac{N_e}{2}$ zero valued DWT coefficients to q_1 and $\frac{N_e}{2}$ zero valued coefficients to q_{-1}. After this modification, the DWT coefficient distribution should theoretically match our model.

Once a value of $\hat{\lambda}$ has been obtained for a DWT subband and the necessary histogram modifications have been performed, we generate the anti-forensic dither which is added to each DWT coefficient. As was the case with anti-forensic dither designed to modify JPEG compressed images, the use of the Laplace distribution to model the distribution of DWT coefficient values in an uncompressed image allows the anti-forensic dither distribution to be expressed using one of two equations. An analogous reduction in the complexity of generating the dither is realized as well, since once again the dither is drawn from only two distributions. The appropriate expression for the anti-forensic dither distribution depends upon the magnitude of the DWT coefficient to which it is added. When modifying nonzero valued DWT coefficients, the anti-forensic dither's distribution is given by

$$P(D = d|Y = q_k, k \neq 0) = \begin{cases} \frac{1}{\alpha_k} e^{-\operatorname{sgn}(q_k)\hat{\lambda} d} & \text{if } (b_k - q_k) \leq d < (b_{k+1} - q_k), \\ 0 & \text{otherwise,} \end{cases} \tag{4.25}$$

where $\alpha_k = \frac{1}{\hat{\lambda}}(e^{-\operatorname{sgn}(q_k)\hat{\lambda}(b_k - q_k)} - e^{-\operatorname{sgn}(q_k)\hat{\lambda}(b_{k+1} - q_k)})$. When modifying zero valued DWT coefficients, the anti-forensic dither's distribution is

$$P(D = d|Y = 0) = \begin{cases} \frac{1}{\alpha_0} e^{-\hat{\lambda}|d|} & \text{if } b_0 > d > b_1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.26}$$

where $\alpha_0 = \frac{1}{\hat{\lambda}}(2 - e^{-\hat{\lambda} b_1} - e^{\hat{\lambda} b_0})$.

Assuming that we accurately estimate our model parameter so that $\hat{\lambda} = \lambda$ and that we accurately correct for truncation effects, the distribution of the anti-forensically modified coefficients in each DWT subband matches the model uncompressed coefficient distribution. Following the framework outlined in (4.6), we can demonstrate this by using the law of total probability as well as (4.19), (4.25), and (4.26) to write

$$\begin{aligned} P(Z=z) &= \sum_k P(Z=z|Y=q_k)P(Y=q_k) \\ &= \sum_{k\leq -1} \tfrac{1}{\alpha_k} e^{\lambda(z-q_k)} \tfrac{1}{2}(e^{\lambda b_{k+1}} - e^{\lambda b_k})\, \mathbb{1}(b_k \leq z < q_{k+1}) \\ &\quad + \tfrac{1}{\alpha_0} e^{-\lambda|z|}(1-\tfrac{1}{2}(e^{\lambda b_0} + e^{-\lambda b_1}))\, \mathbb{1}(b_0 \leq z < b_1) \\ &\quad + \sum_{k\geq 1} \tfrac{1}{\alpha_k} e^{-\lambda(z-q_k)} \tfrac{1}{2}(e^{-\lambda b_k} - e^{-\lambda b_{k+1}})\, \mathbb{1}(b_k \leq z < q_{k+1}) \\ &= \tfrac{\lambda}{2} e^{-\lambda|z|}. \end{aligned} \tag{4.27}$$

Additionally, we can place an upper bound on the absolute distance between an DWT coefficient from an image compressed using a wavelet-based technique and its uncompressed counterpart. For images compressed using SPIHT or related techniques, the addition of anti-forensic dither will not move a DWT coefficient outside of its quantization interval, with the exception of zero-valued coefficients which remain in the interval $[b_{-1}, b_2)$. Since the corresponding uncompressed DWT coefficient must lie in the same interval, the upper bound on this distance becomes

$$|X-Z| \leq \begin{cases} b_{k+1} - b_k & \text{if } k \neq 1, \\ b_2 - b_{-1} & \text{if } k = 0, \end{cases} \tag{4.28}$$

given $b_k \leq X \leq b_{k+1}$. Because JPEG 2000 applies uniform quantization to the coefficient values within each DWT subband, we can use the upper bound given in (4.8) to write

$$|X-Z| \leq Q \tag{4.29}$$

where Q is the quantization step size used to quantize the coefficients that DWT subband.

4.4 Anti-Forensic Blocking Artifact Removal

As was discussed in Section 4.1, if an image is divided into segments during compression, discontinuities are often present across segment boundaries in the decompressed image. These compression fingerprints, known as blocking artifacts, are commonly present in JPEG compressed images and can arise in JPEG 2000 compressed image if tiling is used. Even when blocking artifacts are not visually discernible, they can still be statistically detected [7]. Though the application of anti-forensic dither to an image removes transform coefficient quantization fingerprints, it does not remove blocking artifacts. If a previously compressed image is to be represented as never having undergone compression, these fingerprints must be removed.

While the removal of JPEG blocking artifacts is a well studied problem [25,58], these techniques are designed to remove visible traces of blocking from low to mid quality images. To be successful, an anti-forensic deblocking technique must remove all visual and statistical traces of blocking artifacts without resulting in forensically detectable changes to an image's transform coefficient distributions or introducing new, forensically detectable fingerprints. Because existing deblocking algorithms are not designed to account for these criteria, they are poorly suited for anti-forensic purposes.

In order to remove statistical traces of blocking artifacts, we propose an anti-forensic deblocking technique that operates by first median filtering an image then adding adding low-power white Gaussian noise to each of its pixel values. Letting $u_{i,j}$ and $v_{i,j}$ denote the value of a pixel at location (i, j) in an unmodified image and its deblocked counterpart respectively, our anti-forensic deblocking operation can be expressed as

$$v_{i,j} = \text{med}_s(u_{i,j}) + n_{i,j} \tag{4.30}$$

where $n_{i,j}$ is a zero mean Gaussian random variable with variance σ^2. In this equation, med_s denotes a two dimensional median filter with a square window of size s pixels, explicitly defined as $\text{med}_s(u_{i,j}) = \text{median}\{u_{l,m} | 0 \leq \lfloor \frac{(i-l)}{2} \rfloor \leq s, 0 \leq \lfloor \frac{(j-m)}{2} \rfloor \leq s\}$. We choose to use a median filter instead of a linear lowpass filter because its edge preserving nature tends to result in less visual distortion than simple linear filters. Both the window size of the median filter and the variance of the noise can be tuned according to the strength of the blocking artifacts present in the image. Heavily compressed images require the use of a larger median filter window size and greater noise variance to remove statistical traces of blocking artifacts than lightly compressed images. We compare the anti-forensic performance of this technique to those of existing deblocking algorithms in Section 4.5.3.

4.5 Experimental Results

In order to verify the efficacy of each of our proposed anti-forensic techniques, we have conducted a number of experiments in which we use our anti-forensic techniques to remove compression fingerprints from a set of images, then test each image for evidence of prior compression using several existing forensic techniques. In this section, we present the results of these experiments and analyze the performance of each proposed anti-forensic technique.

4.5.1 JPEG Anti-Forensics

To demonstrate that our anti-forensic DCT coefficient quantization fingerprint removal technique can be used on an image without significantly impacting its visual quality, we show a typical image before and after anti-forensic modification in Figure 4.5. In this figure, the image on the left has undergone JPEG compression using a quality factor of 65 while the image on the right is the JPEG compressed image after anti-forensic dither has been added to its DCT coefficients. No noticeable difference between these images is apparent after visual inspection. This is reinforced by the fact that the PSNR between the two images is 41.63 dB. More importantly, the anti-forensically modified image contains no visual indicators of either previous compression or anti-forensic modification. Since a forensic examiner will not have access to either the unaltered or compressed version of an anti-forensically modified image, these cannot be compared against the anti-forensically modified image. Instead, what is necessary is that the anti-forensically modified image plausibly appears to have never been compressed.

Inspection of the DCT coefficient value distributions of the images shown in Figure 4.5 yeilds similar results. Figure 4.6 shows a histogram of coefficient values

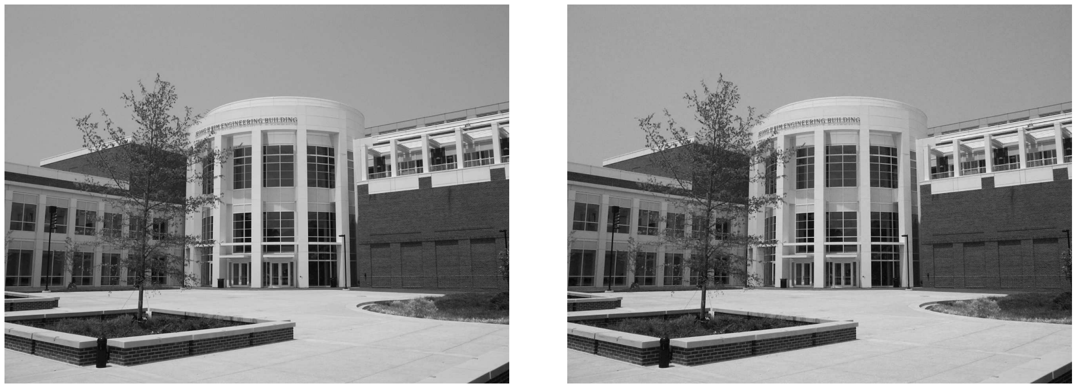


Figure 4.5: Left: JPEG compressed image using a quality factor of 65. Right: Anti-forensically modified version of the same image.

in the (2,2) DCT subband in an uncompressed version of these images along with the corresponding coefficient value histograms from the JPEG compressed and anti-forensically modified images. Figure 4.7 shows the histogram of coefficient values in the DC DCT subband of the same images. While DCT coefficient quantization fingerprints are present in the histograms taken from the JPEG compressed image, these fingerprints are absent in the coefficient value histograms corresponding to the uncompressed and anti-forensically modified images. Again, we note that in reality a forensic examiner will only have access to the anti-forensically modified image and will be unable to make note of minor differences between the coefficient histograms of the uncompressed and anti-forensically modified image. The fact that the DCT coefficient value histograms from the anti-forensically modified image both fit our coefficient distribution model and contain no compression fingerprints suggests that our proposed anti-forensic technique is capable of producing images that can be passed off as never having undergone JPEG compression.

To verify that our anti-forensic technique is able to produce images that can

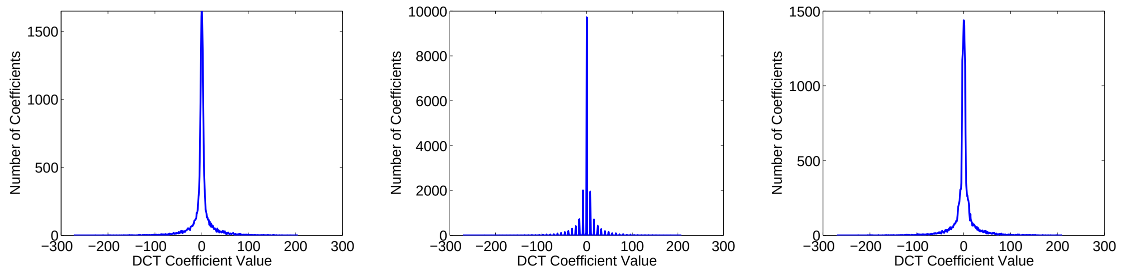


Figure 4.6: Histogram of coefficient values from the (2,2) DCT subband taken from an uncompressed version of the image shown in Fig. 4.5 (left), the same image after JPEG compression (center), and an anti-forensically modified copy of the JPEG compressed image(right).

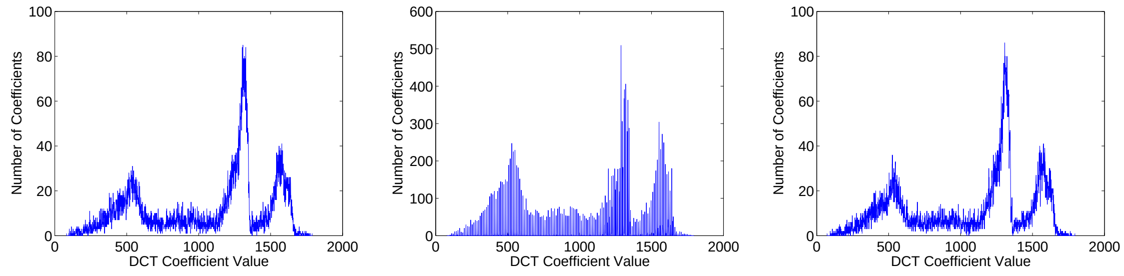


Figure 4.7: Histogram of coefficient values from the DC DCT subband taken from an uncompressed version of the image shown in Fig. 4.5 (left), the same image after JPEG compression (center), and an anti-forensically modified copy of the JPEG compressed image(right).

fool existing forensic compression detection techniques, we conducted the following larger scale experiment. First, we converted each of the 1338 images in the Uncompressed Colour Image Database [38] to grayscale, then we compressed each image using a quality factor of 90, 70, and 50. Next, we removed DCT coefficient quantization fingerprints from the JPEG compressed images by adding anti-forensic dither to the DCT coefficients of each image. Each of the anti-forensically modified images was then tested for DCT coefficient quantization fingerprints using the forensic technique developed by Fan and de Queiroz [7]. This technique detects previous applications of JPEG compression by using the DCT coefficient distributions to estimate the quantization step size used in each DCT subband during JPEG compression. If no evidence of quantization is present in any DCT subband, the image is classified as never-compressed. When we used this technique to search for evidence of JPEG compression in the anti-forensically modified images, it classified each anti-forensically modified image as never-compressed regardless of the quality factor used during compression. These results correspond to a 100% success rate for our anti-forensic DCT coefficient quantization fingerprint removal technique on this data set.

4.5.2 Wavelet Anti-Forensics

We conducted a set of experiments similar to those in Section 4.5.1 on SPIHT compressed images to demonstrate the effectiveness of our anti-forensic DWT coefficient compression fingerprint removal technique. Figure 4.8 shows a typical image compressed at a bitrate of 3.0 bpp using the SPIHT algorithm both before and after anti-forensic dither has been added to its DWT coefficients. As was the case in our JPEG compression anti-forensics example, the two images contain no discernible differences and the anti-forensically modified image shows no signs of compression or anti-forensic modification. Furthermore, the PSNR between these two images is 50.99dB. This result suggests that our anti-forensic DWT coefficient compression fingerprint removal technique will create images containing no visual indicators of compression or anti-forensic modification.

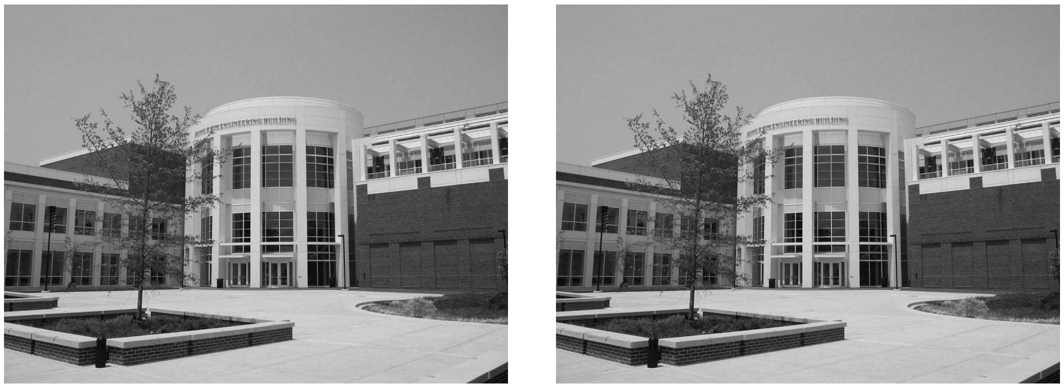


Figure 4.8: Left: An image compressed using the SPIHT algorithm at a bit rate of 3 bits per pixel before the use of entropy coding. Right: The same image after anti-forensic dither has been applied to its wavelet coefficients.

Figure 4.9 shows the DWT coefficient histograms obtained from the fourth level *HH* subband of an uncompressed copy of the image shown in Figure 4.8 as well as from SPIHT compressed and anti-forensically modified versions of the same image. We note that the compression fingerprints observed in the DWT coefficient histogram from the SPIHT compressed image are absent from the DWT coefficient histogram corresponding to the anti-forensically modified image. This, along with the fact that the anti-forensically modified image's DWT coefficient histogram matches our coefficient distribution model, demonstrates that our anti-forensic DWT coefficient compression fingerprint removal technique is capable of modifying images so that they can be passed off as never having undergone wavelet-based compression.

In addition to the experimental results discussed above, we conducted a large scale experiment to demonstrate that our anti-forensic technique is capable of misleading existing forensic wavelet-based compression detection algorithms. To do this, we again converted each of the 1338 images in the Uncompressed Colour Im-

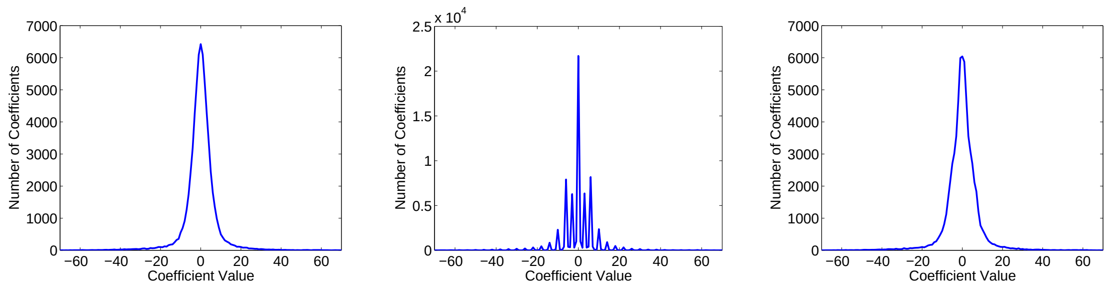


Figure 4.9: Histogram of wavelet coefficients from the fourth level *HH* subband of a four level wavelet decomposition of the image shown in Fig. 4.8 (left), the same image after SPIHT compression (center), and the compressed image after anti-forensic dither has been applied (right).

age Database [38] to grayscale, then compressed them using the SPIHT algorithm at a bitrate of 2.0 bpp. We then removed image compression fingerprints from each image by adding anti-forensic dither to each image's DWT coefficients. Finally, we used the compression detection technique developed by Lin *et al.* [26] to test each image for evidence of prior wavelet-based compression. This detector was trained using the uncompressed and SPIHT compressed images, resulting in a classification rule that was able to correctly identify 99.8% of the SPIHT compressed images while only misclassifying 2.0% of the uncompressed images. When we used the trained wavelet-compression detection algorithm to classify the set of anti-forensically modified images, it was only able to correctly classify 1.2% of them as having undergone compression, resulting in a 98.2% success rate for our anti-forensic DWT compression fingerprint removal technique.

4.5.3 Anti-Forensic Deblocking

To evaluate our anti-forensic deblocking algorithm, we conducted an experiment in which we used it to remove blocking artifacts from several anti-forensically modified images, then compared its performance with those of the JPEG deblocking algorithms proposed by Liew and Yan [58], and Zhai *et al.* [58]. To perform this experiment, we first converted to grayscale and JPEG compressed each of the 1338 images in the Uncompressed Colour Image Database using quality factors of 90, 70, 50, 30, and 10, then applied anti-forensic dither to the DCT coefficients of each of the compressed images. This created a testing database of 6690 anti-forensically modified grayscale images. Next, we used our anti-forensic deblocking algorithm along with the deblocking algorithms proposed by Liew and Yan, and Zhai *et al.* to remove JPEG blocking artifacts from each image.

We tested each of the deblocked images for JPEG blocking fingerprints using the test designed by Fan and de Queiroz [7]. This method operates by collecting two pixel difference measurements throughout an image; one taken at the center of each block, which we refer to as R_1 and a second, which we refer to as R_2, taken across the boundary that occurs at the corners of each set of four adjacent blocks. Next, histograms of the R_1 and R_2 values obtained throughout the image, denoted h_1 and h_2 respectively, are tabulated. Finally, a test statistic K measuring the difference between the two histograms is computed according to the equation

$$K = \sum_r |h_1(r) - h_2(r)|, \qquad (4.31)$$

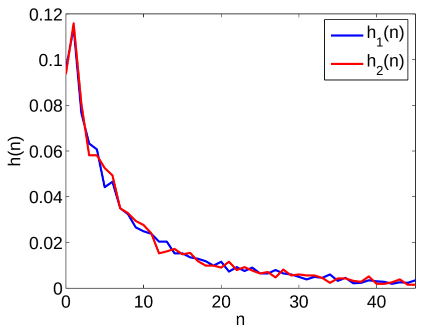
and K is compared to a threshold. If K is greater than the threshold, the image is

	Proposed Method			Liew	Zhai
Quality Factor	$s = 3$, $\sigma^2 = 3$	$s = 3$, $\sigma^2 = 2$	$s = 2$, $\sigma^2 = 2$	& Yan [58]	*et al.* [25]
90	0.0%	0.0%	0.2%	52.5%	98.3%
70	0.0%	0.1%	0.8%	76.3%	96.3%
50	0.2%	0.2%	1.6%	96.8%	96.1%
30	0.3%	10.5%	24.1%	99.2%	93.7%
10	49.0%	79.0%	95.9%	99.0%	70.8%

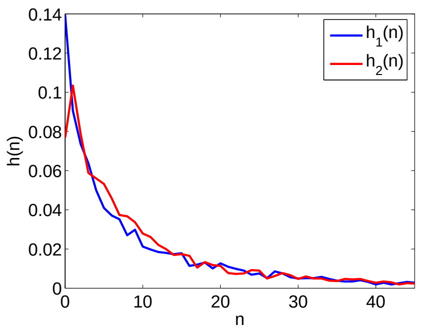
Table 4.1: Blocking artifact detection rates.

classified as one which contains blocking artifacts.

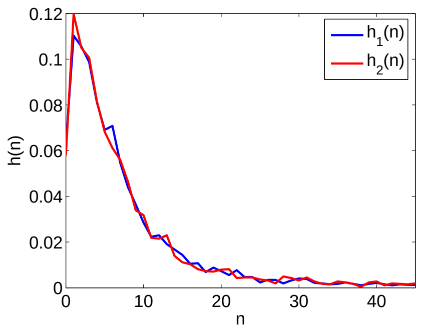
We used the uncompressed and JPEG compressed images from our database to train this forensic blocking artifact detector and selected a decision threshold corresponding to a 99.1% probability of detecting blocking artifacts with a false detection rate of 0.0%. The trained detector was then used to test each of the deblocked images for blocking artifacts. Block artifact detection rates obtained from this experiment are shown in Table 4.1. As we can see from this table, the deblocking methods of Liew and Yan, and Zhai *et al.* are poorly suited for removing statistical trace of blocking fingerprints from compressed images. By contrast, if the parameters s and σ^2 are properly chosen, our proposed algorithm is capable of removing statistical traces of blocking artifacts from images previously JPEG compressed at quality factors of 30 and above.



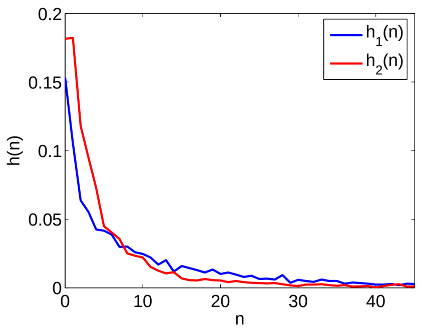
(a)



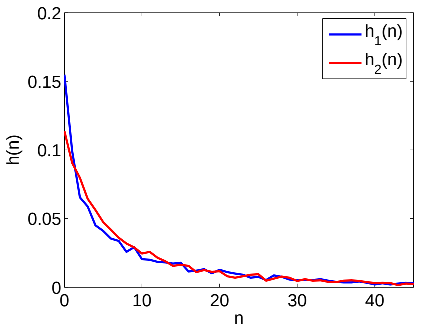
(b)



(c)



(d)

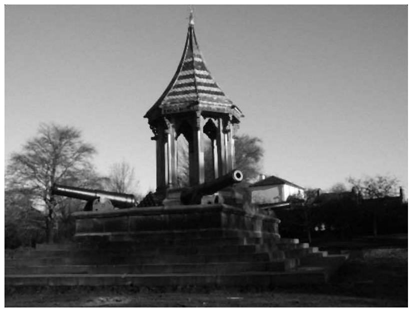


(e)

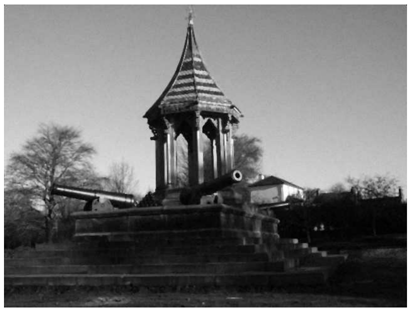
Figure 4.10: Histograms of R_1 and R_2 blocking artifact detection statistics obtained from (a) an uncompressed image, (b) the same image after JPEG compression using a quality factor of 70, as well as the JPEG compressed version after it has been deblocked using (c) our anti-forenic deblocking algorithm, (d) the deblocking algorithm proposed by Liew and Yan, and (e) the deblocking algorithm proposed by Zhai *et al.*.

Additionally, we have discovered that existing deblocking techniques leave behind their own fingerprint. We have observed that under normal circumstances, $h_1(1) < h_1(0)$ and $h_2(1) < h_2(0)$ in an uncompressed image. This can be seen in Figure 4.10 which shows the R_1 and R_2 histograms obtained from a typical image before and after JPEG quantization as well as after the JPEG compressed image was deblocked using our anti-forensic technique and those proposed by Zhai *et al.*, and Liew and Yan. By contrast, $h_1(1) > h_1(0)$ and $h_2(1) > h_2(0)$ in images deblocked using the Zhai *et al.*, and Liew and Yan techniques. This histogram feature can be used as a fingerprint indicating that an image has been deblocked using one of these algorithms. As Figure 4.10 shows, this fingerprint is not present in images modified by our anti-forensic deblocking technique, indicating that it is much better suited for anti-forensic purposes.

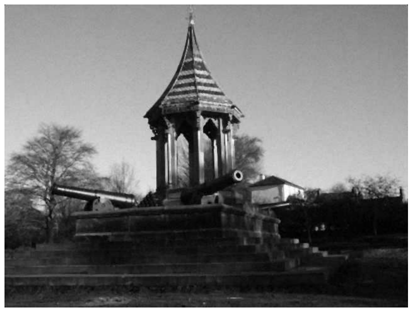
Though our anti-forensic dither and deblocking techniques can successfully remove statistical traces of compression artifacts from heavily compressed images, they cannot compensate for significant visual distortion caused by the initial application of compression. For heavily compressed images, they can serve to significantly increase the distortion present in an image. Figure 4.11 shows a typical image after JPEG compression using several quality factors followed by the anti-forensic removal of both its DCT coefficient quantization fingerprints and its blocking fingerprints. While the images compressed using quality factors of 70 and 90 appear to be unaltered, the images compressed with a quality factor of 30 and below contain noticable distortions. Accordingly, as an image is more heavily compressed, it more difficult to convincingly disguise both visual and statistical traces of its compression history.



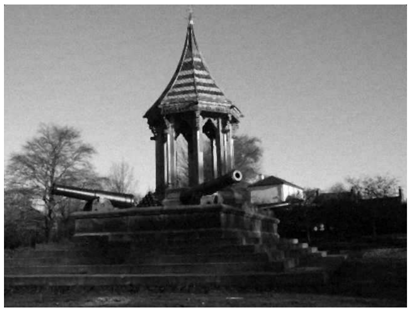
(a)



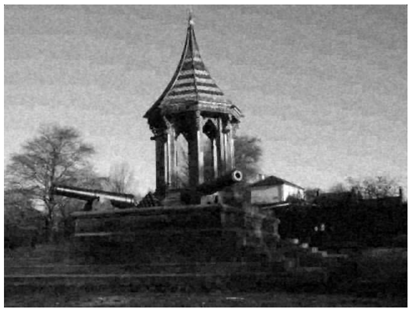
(b)



(c)



(d)



(e)

Figure 4.11: Results of the proposed anti-forensic deblocking algorithm applied to a typical image after it has been JPEG compressed using a quality factor of (a) 90, (b) 70, (c) 50, (d) 30, and (e) 10 followed by the addition of anti-forensic dither to its DCT coefficients.

4.6 Undetectable Image Tampering Using Anti-Forensics

In many scenarios, an image forger is not concerned with representing a previously compressed image as one which has never undergone compression. More likely, the image forger wishes to alter an image, then remove all evidence that the image has been manipulated. In such a scenario, the image forger must pay particular attention to an image's compression history. Because most digital cameras store images as JPEGs by default, many digital images are imprinted with compression fingerprints at the time of their capture. If an image contains evidence that it has been compressed multiple times, this suggest that the image has been decompressed for editing, then saved again in a compressed format. If the forger attempts to avoid the fingerprints left by multiple applications of compression by saving an image in an uncompressed format after editing, the fingerprints left by the initial application of compression will reveal evidence of image manipulation. Furthermore, spatial inconsistencies in an image's compression fingerprints are often used as forensic evidence of cut-and-paste forgery, in which a composite image is formed by cutting an object from one image, then pasting it into another. If used properly, the anti-forensic techniques outlined in this chapter can either remove or prevent the occurrence of each of these image tampering fingerprints.

Recompression of a JPEG image, commonly referred to as double JPEG compression, introduces a unique fingerprint in an image's DCT coefficient distributions. During the initial application of JPEG compression, DCT coefficient quantization causes an image's DCT coefficients to cluster around integer multiples of a particular DCT subband's quantization step size. When the image is compressed a second time using a different quantization table, some DCT subbands will be quantized using a different quantization step size. This mismatch in quantization step sizes will cause an unequal number of DCT coefficient clusters to fall within each new quantization interval. As a result, the DCT coefficient distributions of a double JPEG compressed image will appear to be modulated by a periodic signal. A number of forensic techniques use this signal to identify double JPEG compression [32, 33].

To prevent double JPEG compression fingerprints from occurring in a doubly compressed image, an image forger can add anti-forensic dither to a singly compressed image's DCT coefficients before it is recompressed. By doing this, the image's DCT coefficients will be distributed as if they came from an uncompressed image rather than being clustered around integer multiples of the first quantization step size. When the image is recompressed, quantization interval mismatch effects will not occur, allowing the double JPEG compressed image's DCT coefficients to be distributed as if they came from an image compressed only once. Since the image will remain JPEG compressed in its final state, it does not need to be anti-forensically deblocked.

An example demonstrating that anti-forensic dither can be used to prevent double JPEG compression fingerprints is shown in Figure 4.12. In this example, we show coefficient histograms from the (3,3) DCT subband of an image compressed once using a quality factor of 85, the same image after it has been double compressed using a quality factor 75 followed by 85, as well as the image compressed first with a quality factor 75, then anti-forensically modified and recompressed using a quality

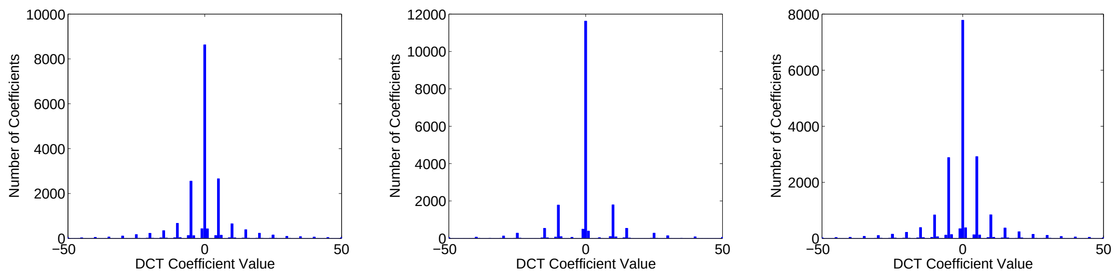


Figure 4.12: Histogram of (3,3) DCT coefficients from an image JPEG compressed once using a quality factor of 85 (Left), the image after being double JPEG compressed using a quality factor of 75 followed by 85 (Center), and the image after being JPEG compressed using a quality factor of 75, followed by the application of anti-forensic dither, then recompressed using a quality factor of 85 (Right).

factor of 85. While double JPEG compression fingerprints can be observed in the coefficient histogram of the doubly JPEG compressed image that did not have anti-forensic dither added to its DCT coefficients, these fingerprints are absent from the coefficient histogram of the image that underwent anti-forensic modification. Additionally, the coefficient histogram of the anti-forensically modified double JPEG compressed image does not differ greatly from the coefficient histogram of the singly compressed image. This verifies that under forensic inspection, the anti-forensically modified image would appear to only have only been compressed once.

If two JPEG compressed images are used to create a cut-and-paste forgery, the composite image will contain double JPEG compression fingerprints that differ spatially. These locally varying fingerprints can be used to both detect forged images and to identify falsified image regions [16]. Alternately, if blocking artifacts in the pasted region do not align with those throughout the rest of the image, the resulting mismatch in the blocking grid can be used to detect cut-and-paste forgeries [57].

Both of these fingerprints can be avoided if the two images used to create the forgery have anti-forensic dither added to their DCT coefficients and are anti-forensically deblocked before the composite image is created. Doing this will render compression history based forensic techniques unable to detect cut-and-paste image forgeries.

In other situations, an image forger may wish to falsify the origin of an image. Since most digital cameras and image editing software use proprietary JPEG quantization tables when storing images, the camera model used to capture an image can be determined by identifying the image's quantization table in a list of camera and quantization table pairings [9]. This means that information about an image's origin is intrinsically embedded in an image via its compression history. Software designed to perform quantization table and camera matching known as *JPEGsnoop* is readily available online [15]. As a result, an image forger cannot mislead forensic investigators by simply changing an image's metadata tags. While other forensic signatures such as a camera's sensor noise [5] and color filter array interpolation parameters [52] can be used as a means of camera identification, these techniques can be defeated by falsifying the sensor noise pattern [12] and by reapplying the color filter array then re-interpolating the image [22] respectively.

An image's origin cannot be forged by simply recompressing it using the quantization table of another camera. Doing this will result in double JPEG compression artifacts that can alert forensic investigators to the fact that the image has been tampered with. Instead, we are able to undetectably falsify the compression history aspects of an image's origin by first removing traces of prior JPEG compression through the use of anti-forensic dither, then compressing the image with the quantization table of another camera.

To verify that our anti-forensic technique is suitable for image origin forgery purposes, we conducted an experiment in which we falsified the compression signatures of images taken by several cameras, then attempted to link each image with its origin using existing forensic techniques. For this experiment, we compiled a database consisting of 100 images from each of the following cameras: a Canon Powershot G7 (Cam 1), Sony Cybershot DSC-W80 (Cam 2), Sony Cybershot DSC-V1 (Cam 3), Fuji Finepix E550 (Cam 4), and an Olympus Camedia C5060 (Cam 5). We removed evidence of prior JPEG compression from each image by adding anti-forensic dither to its DCT coefficients, then recompressed it with the quantization tables used by each of the other cameras in the database. After this was done, we used the procedure developed by Fan and de Quieroz to obtain an estimate $\hat{Q}_{i,j}$ of the quantization table used to compress each image [7]. We matched each image with a camera by selecting the camera whose quantization table $Q_{i,j}^{(k)}$ maximized the similarity measure

$$s_k = \sum_i \sum_j \mathbb{1}(\hat{Q}_{i,j}, Q_{i,j}^{(k)}). \tag{4.32}$$

Table 4.2 shows the results of our image origin forgery experiment. With the exception of representing the images captured by the Sony Cybershot DSC-V1 as originating from the Sony Cybershot DSC-W80, we were able to falsify the origin of the images captured by each camera with a 100% success rate. In the case of the Sony Cybershot DSC-V1, one image was linked to a different camera than the one we intended.

Falsified	True Image Origin				
Origin	Cam 1	Cam 2	Cam 3	Cam 4	Cam 5
Cam 1	-	100.0%	100.0%	100.0%	100.0%
Cam 2	100.0%	-	99.0%	100.0%	100.0%
Cam 3	100.0%	100.0%	-	100.0%	100.0%
Cam 4	100.0%	100.0%	100.0%	-	100.0%
Cam 5	100.0%	100.0%	100.0%	100.0%	-

Table 4.2: Camera origin forgery classification results.

4.7 Summary

In this chapter, we have proposed a set of anti-forensic operations capable of removing compression fingerprints from digital images. To do this, we developed a generalized framework for the removal of quantization fingerprints from an image's transform coefficients. According to this framework, quantization fingerprints can be removed from an image's transform coefficients by first estimating the distribution of the image's transform coefficients before compression, then adding anti-forensic dither to the compressed image's transform coefficients so that their anti-forensically modified distribution matches the estimate of their distribution before compression. We used this framework to design specific anti-forensic techniques to remove DCT coefficient quantization artifacts from JPEG compressed images and DWT coefficient compression artifacts from images compressed using wavelet-based coders. Additionally, we have proposed an anti-forensic technique capable of removing statistical traces of blocking artifacts from images that undergo blockwise segmentation during compresion.

To demonstrate the performance of our algorithms, we have conducted a number of experiments on JPEG and SPIHT compressed images in which we show that by adding anti-forensic dither to an image's transform coefficients, we can render that image's transform coefficient compression fingerprints forensically undetectable without significantly degrading the image's visual quality. We have conducted an experiment showing that our anti-forensic deblocking technique can remove statistical traces of blocking artifacts from images while several existing deblocking techniques cannot. Additinally, we have shown that our proposed anti-forensic techniques can be used to make certain types of image tampering such as double JPEG compression, cut-and-paste image forgery, and image origin falsification undetectable to compression history based forensic techniques.

Chapter 5

Temporal Forensics and Anti-Forensics for Digital Video

Just as digital editing operations leave behind fingerprints, anti-forensic operations may inadvertently leave behind their own fingerprints [49]. If these fingerprints can be identified, forensic techniques can be designed to detect them. This will allow forensic investigators to identify digital forgeries even when editing fingerprints have been anti-forensically removed. Researchers have recently developed techniques to identify anti-forensic manipulation of an image's PRNU [13] and compression history [54].

When confronted with a forensic technique capable of detecting the use of an anti-forensic operation, intelligent forgers will attempt to modify their anti-forensic operation in order to minimize the strength of the fingerprint it leaves behind. This leads to a cat-and-mouse game between a digital forger and a forensic investigator. Furthermore, a digital forger can opt not to completely anti-forensically remove all editing fingerprints left in the forgery. Instead, the forger may decrease the strength of the anti-forensic operation so that it reduces the strength of the editing operations fingerprint to just below a forensic investigator's detection threshold. This will correspondingly reduce the strength of the anti-forensic operations fingerprint, thus helping the attacker avoid detection. The forensic investigator, meanwhile, must ensure that the combination of the false alarm rates from techniques to detect editing and the use of anti-forensics is below a constant false alarm rate.

This interplay between a forensic investigator and a digital forger raises a number of important questions. For example, if a forensic technique is effective at detecting a particular type of forgery but can easily be fooled if a forger makes use of anti-forensics, is it a good or bad detection algorithm? Similarly, if an anti-forensic operation is able to successfully remove fingerprints left by a particular forgery operation but introduces new fingerprints of its own, how do we evaluate its effectiveness? What is the optimal strategy for a forger to use to avoid forensic detection of both their forgery and their use of anti-forensics? What is the optimal detection strategy for a forensic investigator to follow when attempting to identify digital forgeries? Should decision thresholds in forensic detection techniques be chosen to yield the best performance under a worst case scenario, or can knowledge of the attacker's actions be used to improve detection results? Are there certain editing operations that an attacker will be unable to hide both evidence of their manipulation and evidence of their use of anti-forensics?

To address these questions, we analyze the interaction between a digital forger and a forensic investigator in a particular forensic scenario. In this chapter, we consider the problem of forensically detecting video frame deletion or addition. Frame deletion may be performed by a video forger who wishes to remove certain portions of a video sequence, such as a person's presence in a surveillance video. Similarly, a forger may wish to falsify an event by inserting a sequence of new frames into a video segment. In previous work, Wang and Farid demonstrated that frame deletion or addition followed by recompression introduces a forensically detectable fingerprint into MPEG video [55]. Though their detection technique is quite effective, it requires human identification of frame deletion or addition fingerprints and can only be used on videos compressed by a certain class of video encoders that employ a fixed group of picture (GOP) structure.

In this chapter, we propose new video frame deletion or addition forensic and anti-forensic techniques along with a new framework for evaluating the interplay between a forger and forensic investigator [42]. The main contributions of this work can be summarized as follows:

- We propose a mathematical model of video frame deletion and addition fingerprints that show themselves in a video's P-frame prediction error sequence.
- We use this model to develop two new automatic video frame deletion or addition detection technique. One of these techniques is targeted towards video codecs that use fixed length GOPs when compressing a video, while the other is suitable for use with newer compression standards that allow the GOP length to change adaptively.
- We propose an anti-forensic technique capable of hiding frame deletion or addition fingerprints in digital videos. This technique operates by first constructing a target P-frame prediction error sequence that is free from fingerprints, then selectively altering the video's predicted frames so that the prediction error sequence from the anti-forensically modified video matches the target one.
- We identify a new fingerprint that frame deletion or addition anti-forensics introduces into a modified video's motion vectors and propose a forensic scheme

designed to detect it. Additionally, we modify our proposed anti-forensic technique to minimize detection by these means.

- We define a new set of terms to use when evaluating the performance of both forensic and anti-forensic algorithms.

- We propose a set of game theoretic techniques to study the dynamics between a digital forger and a forensic investigator. We do this by formulating each party's utility functions in terms of the probabilistic quantities associated with the performance of their forensic detection technique or anti-forensic operation.

- We use our new techniques to evaluate the forensic and anti-forensic algorithms proposed in this chapter.

The remainder of this chapter is organized as follows. In Section 5.1 we provide an overview of the background material relevant to frame deletion and addition fingerprints, and develop our mathematical model of these fingerprints. In Section 5.2, we use this model to construct a set of automatic frame deletion or addition detection techniques. We propose our anti-forensic technique to remove frame deletion and addition fingerprints in Section 5.3. We then identify the new fingerprints left by this anti-forensic technique, use these fingerprints to develop an algorithm to detect the use of anti-forensics, and modify our proposed anti-forensic technique in response to this in Section 5.4. We discuss the performance evaluation of forensics and anti-forensic algorithms in Section 5.5 and develop our game theoretic techniques to evaluate the dynamics between a forger and forensic investigator. We present the results of several experiments designed to evaluate the performance of each of our proposed techniques in Section 5.6. Finally, we summarize this chapter in Section 5.7.

5.1 Frame Deletion Fingerprints

We begin this section with a brief overview of video compression, with an emphasis on the forensically significant aspects. Next, we discuss prior forensic work on video frame deletion or addition detection. We then propose a new mathematical model of frame deletion and addition fingerprints which we will use to develop our forensic and anti-forensic techniques.

5.1.1 Video Compression Overview

Due to the size of uncompressed digital video files, virtually all digital video undergoes compression during storage or transmission. Though a variety of different video compression techniques exist, the majority operate in the same basic manner. Since a scene typically changes very little over a short period of time, a great deal of redundancy exists between video frames. Video encoders exploit this redundancy by predicting certain frames from others, then storing the prediction error. The prediction error can be compressed at a higher rate than the frame itself, allowing for smaller file sizes.

In order to prevent the propagation of channel and decoding errors, not all frames are predicted. Instead, the video sequence is segmented into sets of frames known as ‘groups of pictures’ (GOPs). Frames are predicted from other frames in the same GOP, but prediction does not occur across GOPs. Within each GOP, frames are assigned one of three types according to the manner in which they are predicted and compressed. These frame types are known as: intra-frames (I-frames), predicted-frames (P-frames), and bidirectional-frames (B-frames).

Each GOP begins with an I-frame. I-frames are not predicted from any other frame and are independently encoded. In video compression standards such as MPEG-1 and MPEG-2, I-frames are encoded using a lossy process nearly identical to JPEG compression. The remainder of each GOP consists of P-frames and B-frames. These frames are predictively encoded using processes known as motion estimation and compensation. A predicted version of the encoded frame is formed from segments of an anchor frame or frames. Only I-frames and P-frames may act as anchor frames.

In MPEG-1 and 2, P-frame motion estimation is performed by first segmenting the frame into 16×16 pixel macroblocks. Next, the preceding anchor frame is searched for the macroblock that best matches each macroblock in the current P-frame. The row and column displacements between each macroblock in a P-frame and its match in the anchor frame are recorded as that macroblock's row and column motion vectors. A motion-compensated, predicted version of the P-frame is formed by assembling each of the matching macroblocks from the anchor frame. The predicted frame is then subtracted from the actual P-frame, resulting in the P-frame's prediction error. This prediction error is compressed using the same JPEG-like process used to encode I-frames.

During storage and transmission, only the motion vectors and prediction errors are retained. To decompress these frames, the predicted version of the P-frame is reformed using its motion vectors and the previous anchor frame, which must be decoded first. Next, the prediction error is decompressed and added to the predicted frame, thus reconstructing the frame. B-frames are encoded in a similar manner, however each macroblock frame can be predicted from the anchor frame that immediately precedes the B-frame, immediately follows the B-frame, or an average of these two predictions can be used.

In MPEG-1, MPEG-2, and similar codecs, the structure of each GOP is fixed, i.e. the sequence of I-, P-, and B-frames always occurs in the same pattern. Newer video compression standards such as MPEG-4 and H.264 allow for the GOP structure to be adjusted depending on the amount of motion in the scene. For example, rapidly changing scenes can be encoded using shorter GOPs because the accuracy of motion compensation greatly decreases as new objects enter each frame.

5.1.2 Detection of Frame Deletion or Addition

In a number of scenarios, a video forger may wish to add or delete frames from a digital video sequence. To do this, the forger must decompress the video before frames are added or deleted, then recompress the video after it has been altered. Previous work by Wang and Farid has shown that recompression of MPEG video using a fixed GOP structure results in two distinct, forensically detectable fingerprints; one spatial and the other temporal [55]. The spatial fingerprint can be observed within a single MPEG I-frame and is similar in nature to the fingerprint

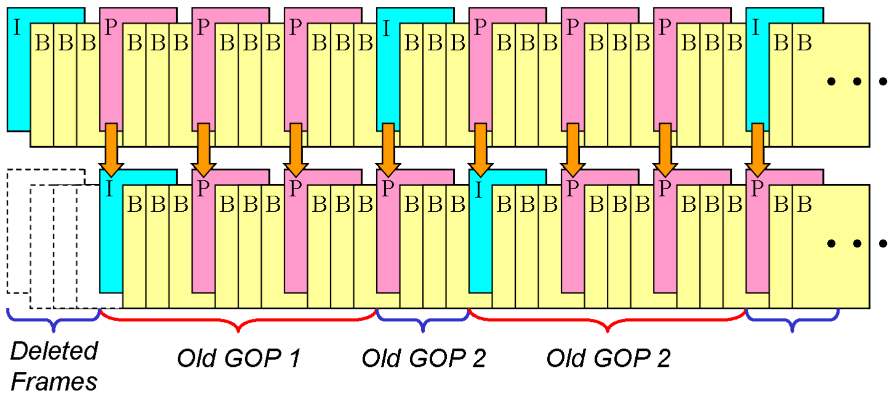

I B B B P B B B P B B B P B B B I B B B P B B B P B B B P B B B I B B
I B B B P B B B P B B B P B B B I B B B P B B B P B B B P B B B P B B
Deleted Frames
Old GOP 1
Old GOP 2
Old GOP 2

Figure 5.1: Illustration of the effects of frame deletion on a video frame sequence. The original video sequence is shown along the top of this figure and the altered video sequence is shown along the bottom. Each GOP in the altered video contains frames from two different GOPs in the unaltered video sequence.

left by double JPEG compression [32, 33]. This fingerprint occurs when either no frames are added or deleted, or when the number of frames added or deleted is an integer multiple of the fixed GOP length. The temporal fingerprint occurs in the sequence of P-frame prediction errors and occurs only if frames have been added to or deleted from the video sequence prior to recompression.

When frames are deleted from or added to a digital video, each GOP in the recompressed video will contain frames that belonged to different GOPs during the initial compression. This effect can be seen in Fig. 5.1, which shows an example of frame deletion for a video compressed using a fixed GOP sequence. Wang and Farid experimentally demonstrated that when a P-frame is predicted from an anchor frame that initially belonged to a different GOP, an increase in the total prediction error is observed [55]. Furthermore, they demonstrated that if a fixed GOP structure is used, this increase in prediction error occurs periodically in the sequence of P-frame prediction errors. As a result, they proposed detecting frame deletion or addition by visually inspecting the sequence

$$e(n) = \frac{1}{N_{xy}} \sum_{x} \sum_{y} |p_{x,y}(n)|, \tag{5.1}$$

for a periodic fingerprint, where N_{xy} is the number of pixels in each frame and $p_{x,y}(n)$ is the prediction error of the n^{th} P-frame at pixel location (x, y) [55]. Alternately, the discrete Fourier transform (DFT) of this sequence $E(k) = \text{DFT}\{e(n)\}$ can be inspected for peaks resulting from the periodic fingerprint. An example of this fingerprint can be seen in Fig. 5.2 which shows the P-frame prediction error sequence of 250 frames of an MPEG-1 compressed version of the commonly used 'Carphone' video, along with the P-frame prediction error sequence of the same video after the first 6 frames have been deleted followed by recompression.

While this frame addition or deletion detection technique is quite successful, it possesses several shortcomings. Because it requires human inspection of the P-frame prediction error sequence or its DFT, Wang and Farid's detection technique can not be run automatically on large amounts of data and is subject to human error. Furthermore, its reliance on human inspection makes it difficult to characterize the performance of this detection technique using a receiver operating characteristic (ROC) curve or other statistical measure. Most importantly, because this detector relies on identifying periodic increases within the P-frame prediction error sequence, it can only be used on videos that are compressed by a codec with a fixed GOP pattern. It cannot be used on videos compressed using more recently developed encoders such as MPEG-4 or H.264 if their implementations adaptively change the

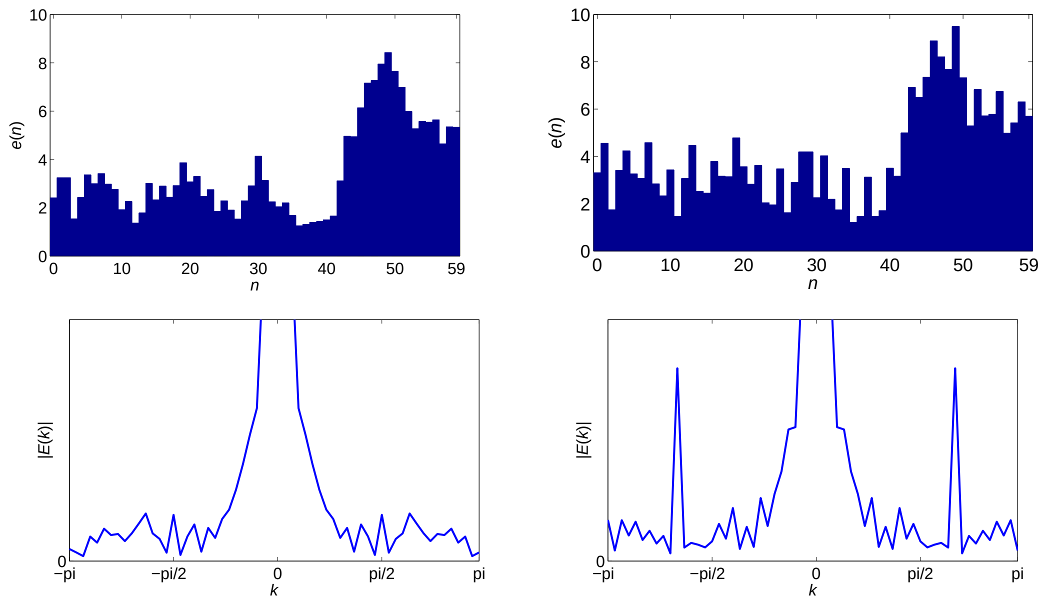


Figure 5.2: P-frame prediction error sequence (top left) and the magnitude of its DFT (bottom left) obtained from an unedited, MPEG compressed version of the ‘Carphone’ video sequence along with the P-frame prediction error sequence (top right) and the magnitude of its DFT (bottom right) obtained from the same video after frame deletion followed by recompression.

GOP length. This is because the increase in the P-frame prediction error will not occur periodically unless a fixed GOP pattern is used.

5.1.3 Temporal Fingerprint Model

In order to design an automatic frame deletion or addition detection technique as well as an anti-forensic method to remove frame addition and deletion fingerprints, we have developed a model of the effect of frame deletion or addition followed by recompression on a video's P-frame prediction error sequence. To simplify our discussion, we will consider only frame deletion for the remainder of this paper. Each of the equations and techniques presented hereafter can be modified to accomodate frame addition by viewing it as the deletion of a negative number of frames.

Let $e_1(n)$ denote the P-frame prediction error sequence of an unaltered video that has been compressed once and let $e_2(n)$ denote the prediction error sequence of that same video after n_D frames have been deleted followed by recompression. We model the relationship between the altered and unaltered videos' P-frame prediction error sequences using the equation

$$e_2(n) = e_1(n - n_D)(1 + s(n)). \tag{5.2}$$

In this equation, the signal $s(n)$ denotes the temporal fingerprint caused by frame deletion. We propose two different models of the temporal fingerprint based on whether the video codec used to perform compression employed a fixed length GOP or an adaptively changing one.

5.1.3.1 Model for Fixed Length GOPs

As was discussed previously, Wang and Farid demonstrated that when using a video codec with a fixed GOP structure frame deletion followed by recompression introduces a periodic trace into a video's P-frame prediction error sequence. Naturally, this leads us to model $s(n)$ in this situation as a periodic signal. The temporal fingerprint's periodicity arises because frame deletion causes a constant shift in the position of each GOP used during the initial compression relative to the locations of the GOPs used during recompression. As a result, each new GOP will contain frames from exactly two GOPs present during the initial application of compression in a repetitive fashion. Using this information and defining T as the period of the temporal fingerprint, we can show that the temporal fingerprint exhibits the following three properties [47]:

Property 1: The temporal fingerprint's repetitive pattern corresponds to a disproportionate increase in $e(n)$ exactly once per fingerprint period.

Property 2: The period T of the temporal fingerprint is equal to the number of P-frames within a GOP.

Property 3: Define the phase ϕ of the temporal fingerprint as the number of P-frames within a GOP before the increase in $e(n)$ due to frame deletion. The phase is determined by the equation $\phi = \lfloor |\mathcal{A}|/n_P \rfloor$, where n_P is the number of P-frames within a GOP, $\mathcal{A}$ is the set of frames at the beginning of each GOP that belonged to the same GOP during the initial application of compression, $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$, and $\lfloor \cdot \rfloor$ denotes the floor operation.

To justify these properties, we note that increases in the P-frame prediction error sequence due to the temporal fingerprint occur when a P-frame is predicted from an anchor frame that belonged to a different GOP during the initial compression. Since each new GOP is comprised of frames from only two GOPs used during the initial application of compression, a P-frame will only be predicted in this manner once per GOP. This justifies the first property. The second property arises because the sequence $e(n)$ consists only of P-frame prediction errors, thus spikes in $e(n)$ due to the temporal fingerprint will be separated by the number of P-frames in a GOP. The third property follows directly from the first two properties. We note that by defining n_G as the number of frames in a GOP and n_F as the number of frames in the video sequence that precede the deleted frames, $|\mathcal{A}|$ is given by the equation $|\mathcal{A}| = n_G - ((n_D + n_F) \bmod n_G)$.

Based on these properties, we model the temporal fingerprint as

$$s(n) = \beta\, \mathbb{1}((n - \phi) \bmod T = 0), \tag{5.3}$$

where $\beta > 0$ and $\mathbb{1}(\cdot)$ denotes the indicator function. This corresponds to modeling the P-frame prediction error sequence of an altered video as a shifted version of the unaltered video's prediction error sequence that is scaled by $(1 + \beta)$ once per fingerprint period.

5.1.3.2 Model for Variable Length GOPs

Newer video compression standards such as MPEG-4 or H.264 allow the GOP length to vary based on the amount of motion in a scene. When frames are deleted from a video then recompressed using one of these codecs, GOPs in the recompressed video will be comprised of frames belonging to multiple different GOPs used during the first compression, but this will not occur in a repeating pattern. Some new GOPs may contain frames from more than two GOPs used during the original compression, while others will contain frames from only one. Nonetheless, frame deletion will alter the GOP which each frame belongs to, but in a random fashion rather than a fixed one. As a result, spikes in the P-frame prediction error sequence occur in a random fashion.

To capture this behavior, we model the P-frame prediction error sequence of a video compressed using variable GOP lengths as

$$s(n) = \beta\,\mathbb{1}(\Theta(n) = 0), \tag{5.4}$$

where $\beta > 0$ is a constant and $\Theta(n)$ is a random variable distributed over the set $\{0, 1\}$. Using this model corresponds to modeling the prediction error sequence of an altered video as a shifted version of the altered version's prediction error sequence with randomly selected values scaled by $(1 + \beta)$.

5.2 Detecting Frame Deletion

To address the weaknesses in Wang and Farid's detection technique, we propose two automatic frame deletion or addition detection techniques; one which exploits the periodic nature of frame deletion fingerprints for fixed GOP length encoders and another suitable for use on videos compressed using variable GOP lengths. We develop these techniques in this section by posing frame deletion detection as a hypothesis testing scenario. We keep the convention that $e(n)$ is the observed P-frame prediction error sequence associated with a video in question, $e_1(n)$ is the prediction error sequence of that video before frames have been deleted, and $e_2(n)$ is the prediction error sequence of the video after frames have been deleted followed by recompression.

Using the convention that the null hypothesis corresponds to the video being unaltered, along with our model from Section 5.1.3, detecting frame deletion can be viewed as differentiating between the following two hypotheses:

$$\begin{aligned} &H_0 : e(n) = e_1(n), \\ &H_1 : e(n) = e_2(n) = e_1(n) + s(n)e_1(n). \end{aligned} \tag{5.5}$$

It is clear from this problem formulation that detecting frame deletion is equivalent to detecting the presence of the term $s(n)e_1(n)$. In order to do this, however, we require some knowledge of what the P-frame prediction error sequence of the unaltered video is. We obtain an estimate of this signal by median filtering the observed prediction error sequence according to the formula

$$\hat{e}(n) = \text{median}\{e(n-1), e(n), e(n+1)\}. \tag{5.6}$$

This estimate has the property that it removes the impulsive spikes in prediction error corresponding to frame deletion fingerprints, while leaving the prediction error sequence of an unaltered video largely intact. We model the relationship between this estimate and the true value of $e_1(n)$ as

$$e_1(n) = \hat{e}(n) + \epsilon(n), \tag{5.7}$$

where $\epsilon(n)$ is a zero mean random variable representing estimation error.

Using this estimate of the unaltered video's P-frame prediction error sequence, we calculate $\hat{s}(n)$, an estimate of the fingerprint signal modulated by the prediction error sequence according to the equation

$$\hat{s}(n) = \max(e(n) - \hat{e}(n), 0). \tag{5.8}$$

If the frame deletion fingerprint is present, $\hat{s}(n)$ will be composed of the modulated fingerprint signal $e_1(n)s(n)$ plus the noise term ϵ. We take the maximum of the difference between $e(n)$ and $\hat{e}(n)$ and zero because we know that the term $e(n)s(n)$ is nonnegative.

Now we can reframe our detection problem as differentiating between the following two hypotheses:

$$\begin{aligned} H_0 &: \hat{s}(n) = \max(\epsilon(n), 0), \\ H_1 &: \hat{s}(n) = \max(s(n)e_1(n) + \epsilon(n), 0). \end{aligned} \tag{5.9}$$

This is equivalent to detecting the presence of the modulated frame deletion fingerprint signal $e_1(n)s(n)$ in noise.

If the video codec used to perform compression uses a fixed GOP structure, we are able to leverage the periodic nature of $s(n)$ when performing detection. Because the number of P-frames in one GOP can be determined from the encoded video, the detector can assume knowledge of the fingerprint's period. The phase, however, is unknown to the detector because it depends on information (the number of frames deleted and the point in the video sequence at which frame deletion occurs) that is hidden from the forensic investigator. As a result, fingerprint detection is well suited for the frequency domain, where the presence of a periodic signal can be readily determined without requiring information about its phase.

To perform frame deletion detection when the video codec uses a fixed GOP structure, we first calculate $\hat{S}(k) = |\,\text{DFT}\{\hat{s}(n)\}|$, the magnitude of the DFT of the video in question's P-frame prediction error sequence. For a prediction error sequence N frames long, a peak will occur in $\hat{S}(k)$ at $k^* = N/T$ if frame deletion fingerprints are present. We measure the strength of this peak using the detection statistic ρ, defined as

$$\rho = \frac{\hat{S}(k^*)}{\sum_{k\in\Omega} w(k)\hat{S}(k)} \tag{5.10}$$

where $w(k) = ce^{-\lambda(k-k^*)^2}$ and $\Omega = \{k | k \leq N/2, k \neq 1, k^*\}$. The function $w(k)$ is used to weight $\hat{S}(k)$ values closer to k^* more than those further away. The variable c is a normalizing constant chosen such that $\sum_{k\in\Omega} w(k) = 1$.

We have observed that for videos with very low average prediction error levels, the total prediction error for P-frames predicted from I-frames is slightly more than for P-frames predicted from other P-frames. By requiring videos with very low average prediction error powers to exhibit stronger periodic fingerprints as evidence of frame deletion, we are able to reduce the number of false alarms. We detect frame deletion using the following decision rule:

$$\delta_{fixed} = \begin{cases} H_0 & \text{if } \rho e^{\gamma e_{avg}} < \tau_{fixed} \\ H_1 & \text{if } \rho e^{\gamma e_{avg}} \geq \tau_{fixed}, \end{cases} \tag{5.11}$$

where τ_{fixed} is a decision threshold, γ is a scalar constant, and e_{avg} is the average of the prediction error sequence $e(n)$.

If the video is compressed using a newer video compression standard that uses variable GOP lengths, the frame deletion fingerprint will not be periodic. In this case, frame deletion detection is equivalent to detecting an unknown signal in the presence of noise. As a result, we use an energy detector to identify the presence of $s(n)$. This yields the following decision rule

$$\delta_{var} = \begin{cases} H_0 & \text{if } \frac{1}{N}\sum_{n=1}^{N} |\hat{s}(n)| < \tau_{var} \\ H_1 & \text{if } \frac{1}{N}\sum_{n=1}^{N} |\hat{s}(n)| \geq \tau_{var}. \end{cases} \tag{5.12}$$

where τ_{var} is a decision threshold. While the periodicity based decision rule δ_{fixed} cannot be used on videos compressed with variable GOP lengths, the energy detector based decision rule δ_{var} can be used on any video.

5.3 Frame Deletion Anti-Forensics

To undetectably delete a sequence of frames from a digital video, a forger must remove frame deletion fingerprints from the video's P-frame prediction error sequence. The forger is constrained, however, in how they accomplish this. Any anti-forensic technique designed to accomplish this must not introduce an unacceptable amount of distortion into the anti-forensically modified video. Furthermore, the anti-forensically modified video must be decodable by standard video decoders.

In order to develop an anti-forensic technique to remove frame deletion fingerprints, let us first examine how a video's prediction error sequence can be manipulated. Each frame's prediction error is dependent on the accuracy of the predicted version of that frame. Normally, video encoders attempt to create highly accurate predictions of each frame so that the total prediction error is minimized. This reduces the size of the compressed video file. If a less accurate prediction technique is used, the total prediction error for a frame increases. In fact, any total prediction error value associated with a valid frame prediction is achievable. This implies that the total prediction error for a frame can be increased by purposefully choosing motion vectors that yield a poor predicted frame. We note that doing this does not introduce new distortion into the video since each frame can still be recovered by reconstructing its predicted version from the set of encoded motion vectors, then adding the prediction error to the predicted frame.

Using this information, we propose an anti-forensic technique that operates roughly as follows. First, we construct a target P-frame prediction error sequence $\tilde{e}(n)$ that is free from frame deletion fingerprints. Next, we increase the prediction error for each P-frame until the target prediction error is reached. We do this by selectively setting the motion vectors for certain macroblocks to zero, then recalculating the prediction error associated with that macroblock. By modifying the video in this way, we are able to meet both of the previously mentioned criteria imposed upon the anti-forensic technique.

When constructing the target prediction error sequence, we must ensure that it is achievable. Since we can only increase the prediction error, this implies that $\tilde{e}(n) \geq e_2(n)$. Nonetheless, we still wish to keep the prediction error as small as is reasonably possible. With this in mind, we construct our target prediction error sequence by setting $\tilde{e}(n) = e_2(n)$ for values of n for which $e_2(n) = (1+\beta)e_1(n)$. If the encoder used to compress the video employs a fixed GOP structure, this will correspond to n values such that $(n-\phi) \bmod T = 0$. Otherwise, these n values can be identified by comparing the GOP sequence of the unaltered video to the GOP sequence used during recompression. We determine the remaining values of $\tilde{e}(n)$ by interpolating them using a cubic spline. This ensures that no frame deletion fingerprints will occur in the target P-frame prediction error sequence.

After we have generated the target prediction error sequence, we must modify the motion vectors and prediction errors of each P-frame so that the actual P-frame prediction error matches the target error. Since we chose $\tilde{e}(n) = e_2(n)$ for values of n where $e_2(n) = (1+\beta)e_1(n)$, we do not need to modify these P-frames. For the remaining P-frames, we determine the increase in the prediction error incurred by each macroblock if its motion vectors are set to zero. We then zero out the motion vectors of the macroblocks whose prediction error increases the least until the target prediction error level is reached. An explicit description of this procedure is provided below.

Let $b_{i,j}(n)$ denote of sum of the absolute value of the prediction error in the macroblock at location (i,j) in the n^{th} P-frame when motion prediction is used and let $\hat{b}_{i,j}(n)$ be the sum of the absolute value of the prediction error in the same location when the macroblock's motion vector has been set to zero. We define the increase in the macroblock's prediction error caused by setting its motion vector to zero as

$$q_{i,j}(n) = \hat{b}_{i,j}(n) - b_{i,j}(n). \tag{5.13}$$

We note that $q_{i,j}(n) \geq 0$ because the zero motion vector is included in the search space for the optimal motion vector during compression.

Next, we define $\mathcal{Q}^{(l)}(n)$ as the set of indices of the macroblocks that result in the l smallest prediction error increases when their motion vectors are set to zero. More explicitly, $\mathcal{Q}^{(l)}(n)$ is defined as

$$\mathcal{Q}^{(l)}(n) = \left\{(i,j)|q_{i,j}(n) \leq q^{(l)}(n)\right\}, \tag{5.14}$$

where $q^{(l)}(n)$ is the l^{th} smallest entry of $q(n)$.

The total absolute prediction error $g_n(l)$ in the n^{th} frame that results from setting the motion vectors of each macroblock whose indices are in $\mathcal{Q}^{(l)}(n)$ to zero is given by the equation

$$g_n(l) = \sum_{(i,j)\in\mathcal{Q}^{(l)}(n)} \hat{b}_{i,j}(n) + \sum_{(i,j)\notin\mathcal{Q}^{(l)}(n)} b_{i,j}(n). \tag{5.15}$$

The value of l that minimizes the absolute distance between the target prediction error level and the actual prediction error level is

$$l^* = \arg\min_l |g_n(l) - \hat{e}(n)|. \tag{5.16}$$

To remove the temporal fingerprint from the n^{th} P-frame of the recompressed video, we set the motion vectors of each macroblock whose indices are in $\mathcal{Q}^{(l^*)}(n)$ to zero, then recompute the prediction error at these macroblock locations during recompression. Due to the relatively small number of macroblocks in each frame, we find l^* for each frame through an exhaustive search.

In some instances, the target prediction error value for a particular P-frame is greater than the error incurred by setting all of the frame's motion vectors to zero. If this is the case, we search first for the set of motion vectors that maximize the prediction error associated with each macroblock. Because many decoders place a limit on the maximal length of each motion vector, this search must be conducted over the set of allowable motion vectors for a given codec. We increase the frame's prediction error by changing several of its motion vectors to these new, maximal error motion vectors rather than by setting them to zero. The rest of our anti-forensic technique remains the same.

5.4 Detecting the Use of Frame Deletion Anti-Forensics

In the introduction to this paper, we discussed the possibility that anti-forensic operations may leave behind new fingerprints of their own. In this section, we show that this is true for the case of frame deletion and addition anti-forensics.

In order to remove frame deletion fingerprints from the P-frame prediction sequence of a video, that video's motion vectors must be altered in order to increase the prediction error. Despite this, the true motion present in the video does not change. As a result, there is a discrepancy between many of the motion vectors stored in an anti-forensically modified video and the true motion of that video scene. This is not the case for an unaltered video because normal video encoders will attempt to estimate scene motion as accurately as possible in order to minimize each frame's prediction error. Accordingly, these discrepancies between a video's stored motion vectors and the actual motion of the scene are fingerprints left by frame deletion anti-forensics.

To detect the use of frame deletion anti-forensics, we propose comparing a compressed video's P-frame motion vectors to an estimate of the true motion present in the video scene. We accomplish this by first decompressing the video in question, then performing motion estimation on the video to obtain a new set of row and column motion vectors. When estimating the true motion of the video, we use an exhaustive search to determine each motion vector. We note that care must be taken to ensure that each frame is predicted from the same anchor frame used by the compressed video.

Let $r_{u,v}(n)$ and $c_{u,v}(n)$ denote the stored row and column motion vectors at macroblock location (u, v) in the n^{th} P-frame of a compressed video whose authenticity is questioned. Similarly, let $\hat{r}_{u,v}(n)$ and $\hat{c}_{u,v}(n)$ denote the row and column motion vectors estimated from the decompressed video. We compute the mean squared Euclidean distance d between the stored and estimated motion vectors at each frame as

$$d(n) = \frac{1}{UV} \sum_{u=1}^{U} \sum_{v=1}^{V} (r_{u,v}(n) - \hat{r}_{u,v}(n))^2 + (c_{u,v}(n) - \hat{c}_{u,v}(n))^2, \tag{5.17}$$

where U and V are the number of row and column macroblocks in each video frame.

Since not every frame requires anti-forensic modification to raise its error level to the anti-forensic target level, some frames will have distinctly larger $d(n)$ values than others. As a result, a signal similar to the fingerprint signal $s(n)$ occurs in $d(n)$ for anti-forensically modified videos. We exploit this information by measuring the strength of this periodic signal as $d_{freq} = D(k^*)$ where $D(k) = \text{DFT}\{d(n)\}$, and $k^* = N/T$ as defined in Section 5.2. Additionally we obtain a measure $d_{mean} = \frac{1}{n}\sum_{n=1}^{N} d(n)$ of the mean $d(n)$ value. We combine both of these features into a feature vector $\mathbf{d} = [d_{mean}\ d_{freq}]$, then use prinicpal component analysis to reduce its dimensionality to a one dimensional feature d_α.

Framing the detection of frame deletion anti-forensics as a hypothesis testing problem, we adopt the convention that the null hypothesis (H_0) is that the video has not undergone anti-forensic modification and the alternative hypothesis (H_1) is that the video has been anti-forensically modified. We detect the use of frame deletion anti-forensics using the following decision rule:

$$\delta_{mv} = \begin{cases} H_0 & \text{if } d_\alpha < \tau_{mv} \\ H_1 & \text{if } d_\alpha \geq \tau_{mv}, \end{cases} \tag{5.18}$$

where τ_{mv} is the decision threshold.

If a forensic investigator is aware of the possibility that anti-forensics have been used, we must assume that a digital forger will be aware of techniques designed to detect their use of anti-forensics. Since we detect the use of frame deletion anti-forensics by analyzing a video's motion vectors, an intelligent forger will modify the anti-forensic algorithm in an attempt to minimize the mean squared Euclidean distance between the anti-forensically modified motion vectors and the true scene motion.

Because the mean squared Euclidean distance is used to compare a video's motion vectors to an estimate of its true motion, large differences between the anti-forensically modified motion vectors will be penalized more than small differences. This is reasonable because while small errors might realistically occur during motion estimation, large motion estimation errors are far less likely. Naively setting the motion vectors of several macroblocks to zero has the potential to create sizable disparities between these motion vectors and the macroblock's true motion. If the target prediction error can be achieved by introducing small changes to a large set of motion vectors rather than large changes to a small set, the mean squared Euclidean distance between the anti-forensically modified motion vectors and the true motion will be reduced. This will correspondingly decrease the probability that the use of anti-forensics is detected. In light of this, we perform the following modifications to our proposed anti-forensic technique.

Rather than increasing a P-frame's prediction error by setting several of its motion vectors to zero, we instead fix a search radius with an initial value of one pixel around each true motion vector. We then search the set of motion vectors lying inside these search radii for the set of motion vectors that maximize the total prediction error. If the target prediction error is not achievable using motion vectors within the current search radius, the search radius is incremented by one pixel and the search is repeated. This process is iterated until the target prediction error is achievable at a particular search radius.

Once the appropriate radius is determined, the new anti-forensic motion vectors and prediction errors are determined using a process similar to that propose in Section 5.3. The only modification required is that we change $\hat{b}_{i,j}(n)$ to be the sum of the absolute value of a macroblock's prediction error when that macroblock's motion vectors are anti-forensically obtained using the final search radius. Similarly, we change $b_{i,j}(n)$ to the macroblock's total prediction error when the macroblock's motion vectors are anti-forensically determined using the final search radius minus one.

5.5 Performance Analysis and Tradeoff

While digital forensic techniques have been studied for roughly a decade, anti-forensic techniques are relatively new. Presently, few tools exist to evaluate the performance of anti-forensic techniques. Still fewer tools exist to understand the optimal set of actions of a forensic investigator and forger when the forger's use of anti-forensics can be detected. In this section, we propose new techniques for evaluating the performance of an anti-forensic operation. Additionally, we propose a game theoretic framework for analyzing the interplay between a forensic investigator and a forger [41].

5.5.1 Evaluating the Performance of Anti-Forensic Techniques

Let ψ be a digital multimedia file and $m(\cdot)$ be an editing operation capable of manipulating ψ. In order to verify the authenticity of ψ, a forensic investigator will attempt to determine if ψ is actually a manipulated version of another, unaltered digital multimedia file ψ'. This forensic manipulation detection problem can be formulated as differentiating between the following two hypotheses:

$$\begin{aligned} H_0 &: \psi \neq m(\psi'), \\ H_1 &: \psi = m(\psi'). \end{aligned} \tag{5.19}$$

To identify the correct hypothesis, the forensic investigator will employ a detection algorithm δ_m designed to detect the use of m by measuring the strength of its fingerprints in ψ. Typically, this is done by calculating a detection statistic and comparing it to a decision threshold. The decision threshold is chosen to maximize the detection algorithm's probability of detection, defined as $P_d(\delta_m) = P(\delta_m = H_1 | \psi = m(\psi'))$, without violating a constraint on its probability of false alarm, defined as $P_{fa}(\delta_m) = P(\delta_m = H_1 | \psi \neq m(\psi'))$. We adopt the convention that $\delta_m^{(P_{fa})}$ specifies the detection algorithm δ_m operating using the decision threshold associated with the false alarm rate P_{fa}.

Once a detection technique has been established, a digital forger can create an anti-forensic technique α_m designed to fool δ_m. In the past, the performance of an anti-forensic technique has often been measured by the probability that δ_m will identify a multimedia file as unaltered when it has actually been edited using m then anti-forensically manipulated using α_m, i.e. $P(\delta_m(\alpha_m(\psi)) = H_0 | \psi = m(\psi'))$. This measure is biased, however, because a file altered using m will not be identified as manipulated with probability $1 - P_d$ even if anti-forensics are not used. As a result, this measure unfairly credits a number of missed detections to the effects of anti-forensics, thus overestimating the performance of α_m.

Instead, the performance of an anti-forensic technique should be measured in terms of its *anti-forensic effectiveness*, or its ability to cause the missed detection of an altered multimedia file given that the manipulation is detectable if anti-forensics are not used. As a result, we define the probability of anti-forensic effectiveness of α_m as

$$P_{ae}(\alpha_m) = P(\delta_m(\alpha_m(\psi)) = H_0 | \, \delta_m(\psi) = H_1, \psi = m(\psi')). \tag{5.20}$$

It is important to note, however, that an anti-forensic operation need not achieve a $P_{ae} = 1$ in order to render δ_m ineffective. In fact, α_m only needs to cause δ_m to

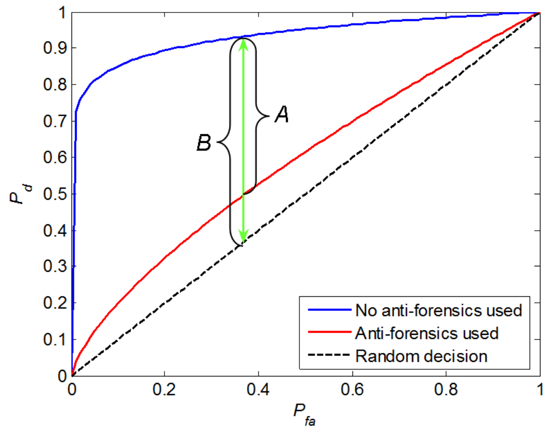


Figure 5.3: Example relating the anti-forensic effectiveness of an anti-forensic operation to the ROC curves achieved by a forensic technique when anti-forensics is and is not used. The anti-forensic effectiveness at a given false alarm level is the ratio A/B.

miss a sufficient number of detections for its performance to become equivalent to making a random decision, or in other words $P_d(\delta_m^{(P_{fa})}) = P_{fa}$. In light of this, it is important to measure the degree to which a forensic technique is susceptible to an anti-forensic attack. As a result, we define the *anti-forensic susceptibility* of a forensic detection technique δ_m operating with a false alarm constraint of P_{fa} to an anti-forensic attack α_m as

$$S_\alpha(\delta_m, P_{fa}) = \frac{P_d(\delta_m^{(P_{fa})}) - \max(P_d(\delta_m^{(P_{fa})})(1 - P_{ae}(\alpha_m)), P_{fa})}{P_d(\delta_m^{(P_{fa})}) - P_{fa}}. \tag{5.21}$$

At a particular false alarm level, the numerator of S_α is the difference between the probability that δ_m will detect manipulation if anti-forensics is not used and the probability that δ_m will detect manipulation if anti-forensics is used to disguise manipulation fingerprints. More explicitly, it is the decrease in the performance of δ_m due to the use of α_m, as shown by the distance A in Fig. 5.3. When computing this distance, we take the maximum between probability that δ_m will detect manipulation if anti-forensics is used, i.e. $P_d(\delta_m^{(P_{fa})})(1-P_{ae}(\alpha_m))$, and the probability of false alarm because the forensic investigator can always achieve $P_d = P_{fa}$ by randomly deciding that a multimedia file is manipulated with probability P_{fa}. Any decrease in the performance of δ_m beyond this point is unnecessary to render δ_m ineffective.

To normalize S_α, its denominator is the difference between the probability of detection achieved by $\delta_m^{(P_{fa})}$ and its corresponding false alarm rate. This difference, which corresponds to the distance B shown in Fig. 5.3, is the maximum decrease in the performance of the forensic detection technique that an anti-forensic attack can cause. As a result, the anti-forensic susceptibility is a measure between 0 and 1 of the decrease in the effectiveness of δ_m caused by α_m. An anti-forensic susceptibility of one indicates that α_m is able to cause δ_m to perform no better than a random decision, while an anti-forensic susceptibility of zero signifies that α_m is unable to cause any reduction in the performance of δ_m. We note that S_α is undefined for $P_{fa} = 1$ because under this condition, no anti-forensic technique is able to cause any reduction in the performance of the forensic detector (it will always decide that the file has been manipulated).

5.5.2 Analysis the Interplay Between a Forger and Forensic Investigator Using Game Theory

In many instances, an anti-forensic operation will leave behind forensically detectable fingerprints of its own. If this is the case, a new forensic detection technique δ_α can be designed to detect the use of α_m. Under this scenario, the forensic detector must determine whether a digital multimedia file is a manipulated and anti-forensically modified version of another unaltered file or not. This problem can be framed as a hypothesis test by defining the two hypotheses as

$$\begin{aligned} H_{0\alpha} &: \psi \neq \alpha_m(m(\psi')), \\ H_{1\alpha} &: \psi = \alpha_m(m(\psi')). \end{aligned} \tag{5.22}$$

To avoid confusion, we rename the previous hypotheses used in the manipulation detection scenario as H_{0m} and H_{1m}. By formulating the detection of anti-forensic manipulation in this manner, the performance of δ_α can be measured using the probabilities of detection and false alarm as before.

The existence of a detection technique capable of identifying the use of anti-forensics poses a new problem for a forger: should anti-forensics be used to disguise a forgery if the use of anti-forensics can itself be detected? A multimedia file will be identified as forged if either manipulation or the use of anti-forensics is detected, therefore a forger must attempt to hide evidence of both. In response, the forger may design the anti-forensic operation in such a way that the strength with which it is applied can be adjusted. By reducing the strength of the anti-forensic attack, a forger decreases the strength of fingerprints left by anti-forensics and correspondingly decreases the probability that the use of anti-forensics will be detected. This is not without a cost, however, because as the strength with which anti-forensics is applied is decreased, the strength of manipulation fingerprints remaining in a multimedia file will increase. This will correspond to an increase in the probability that manipulation will be detected. As a result, the forger must identify the strength with which to apply the anti-forensic operation that minimizes the probability that either the manipulation of the multimedia file or the use of anti-forensics will be detected.

Additionally, some anti-forensic operations degrade the quality of the digital multimedia file that they are used on. If this occurs, it is possible that a human inspecting a forgery may be able to perceptually identify the forgery even if it does not contain detectable manipulation or anti-forensic fingerprints. Alternately, a human may not be able to determine that the multimedia file has been forged, but the perceptual quality of the forgery may be so low that it is rendered useless. In these cases, the forger must also take the perceptual quality of their forgery into account when choosing the appropriate anti-forensic strength.

It is fairly obvious that the optimal anti-forensic strength for the forger to use depends on the decision thresholds used by δ_m and δ_α. Consequently, a forensic detector will choose the decision thresholds for both δ_m and δ_α that maximize the probability of detecting a forgery. Typically, however, a forensic investigator is not free to choose any set of decision thresholds because of a false alarm constraint. Since the probabilities of false alarms associated with both δ_m and δ_α contribute to the total probability of false alarm, the forensic detector must decide how much to allow each detection technique to contribute to the total probability of false alarm. This implies that the probability of false alarm allocation that maximizes the forensic detector's probability of detecting a forgery depends on the anti-forensic strength used by the forger. As a result, both the forger and forensic investigator's optimal actions depend on the actions of their counterpart.

The dependence of the forensic investigator's and forger's optimal actions on the actions of the other naturally leads to the following question; is there a set of actions (i.e. anti-forensic strength and probability of false alarm allocation) for the both the forger and forensic investigator that neither have any incentive to deviate from? Furthermore, if this set of actions exists and both parties take these actions, what is the probability that a forgery will be detected? To answer these questions we use game theory to evaluate the dynamics between the forensic investigator and the forger.

To formulate this situation as a game, we let player 1 denote the forensic investigator and player 2 denote the forger. We adopt the convention that player 1 moves first, or in other words, the forensic investigator chooses the probability of false alarm allocation and corresponding decision thresholds first, then allows the forger to respond. Given a total probability of false alarm constraint ξ, the set of strategies that the forensic investigator can employ is the set of false alarm levels $\eta \in [0, \xi]$ that can be allocated to δ_m. The corresponding false alarm level $\tilde{\eta}$ allocated to δ_α is the maximum false alarm level such that $P_{fa}^{Tot} = P\left(\delta_m^{(\eta)}(\psi) = H_{1m} \bigcup \delta_\alpha^{(\tilde{\eta})}(\psi) = H_{1\alpha} | \psi \neq m(\psi'), \psi \neq \alpha_m(m(\psi'))\right) \leq \xi$. Let $\alpha_m^{(k)}$ be an anti-forensic operation operating at strength $k \in [0, 1]$, where $k = 1$ corresponds to using anti-forensics at full strength and $k = 0$ is equivalent to not using anti-forensics at all. The set of strategies that the forger can employ is the set of anti-forensic strengths $k \in [0, 1]$.

For a particular pairing of strategies (η, k), the utility of player 1 is the probability that either manipulation or the use of anti-forensics will be detected, i.e.

$$U_1(k,\eta) = P\left(\delta_m^{(\eta)}(\psi) = H_{1m} \bigcup \delta_\alpha^{(\tilde{\eta})}(\alpha_m^{(k)}(\psi)) = H_{1\alpha} | \psi = m(\psi')\right). \tag{5.23}$$

Because this corresponds to the probability that a forgery will be detected, player 1 wishes to maximize this utility. By contrast, player 2 wishes to minimize this quantity along with some measure $\gamma(m(\psi), \alpha_m^{(k)}(m(\psi)))$ of the perceptual distortion introduced into the forgery by the use of anti-forensics. As a result, the utility of player 2 is

$$U_2(k,\eta) = -U_1(k,\eta) - \gamma\Big(m(\psi), \alpha_m^{(k)}(m(\psi))\Big). \tag{5.24}$$

By substituting in the appropriate expressions for the probabilistic quantities in each utility function, we can find the Nash equilibrium strategies (η^*, k^*) that neither player has an incentive to deviate from. In practice, however, the analytical evaluation of these utilities is often difficult or impossible. In many forensic scenarios, no known equation exists to express the probabilistic quantities used in each utility function. As a result, the Nash equilibria must often be sought out numerically.

Once the Nash equilibrium strategies have been identified, we can evaluate the probability that the forensic investigator will detect a forgery. To do this, we simply need to evaluate $U_1(\eta^*, k^*)$ because this probability is the utility of player 1. Since the strategy of player 1 is influenced by the false alarm constraint ξ placed on the forensic investigator, it is possible that different Nash equilibrium strategies and different probabilities of forgery detection will be achieved at different ξ levels. By varying ξ between 0 and 1, we can determine the probability of detecting a forgery at the Nash equilibrium associated with each ξ value. Using this information, we can construct a receiver operating characteristic (ROC) curve that displays the forensic investigator's ability to detect a forgery at each false alarm level if both players act rationally. We call this ROC curve the Nash equilibrium receiver operating characteristic (NE ROC) curve. It is this curve, rather than the individual ROC curves of each forensic detection technique, that most accurately characterizes a forensic investigator's ability to detect a digital forgery.

5.6 Experiments and Results

We have conducted a series of experiments to evaluate the performance of each of our proposed forensic and anti-forensic techniques. In order to create data suitable for our experiments, we compiled a set of 36 standard video test sequences in the QCIF format (i.e. a frame size of 176×144 pixels). A complete list of the names of these sequences, along with information regarding where these sequences can be downloaded, is provided in the Appendix. Because these sequences are distributed in an unaltered and uncompressed state, we were able to completely control each video's processing history and ensure that no fingerprints left by other signal processing operations affected our experimental results.

Next, we simulated MPEG-2 compression and decompression in Matlab. In this implementation, we used a fixed twelve frame GOP structure *IBBPBBPBBPBB* along with the standard MPEG-2 DCT coefficient quantization tables. During motion estimation, we determined the set of motion vectors for a predicted frame using

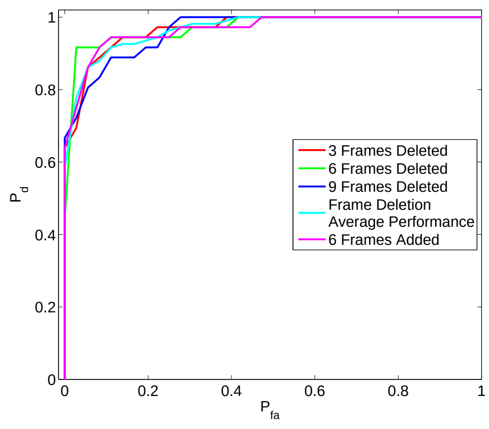


Figure 5.4: ROC curves for δ_{fixed} obtained by testing against different amounts frame deletion and addition.

an exhaustive search. We then compressed the first 250 frames of each uncompressed video sequence, creating a database of unaltered videos compressed using a fixed GOP length.

Because newer compression schemes such as H.264 allow the GOP structure to vary during encoding, we modified our encoder so that it randomly chose between the GOP structures *IBBPBB*, *IBBPBBPBB*, and *IBBPBBPBBPBB* for each GOP during encoding. By doing this, we were able to simulate the forensically significant manner in which H.264 differs from MPEG-2: its use of variable GOP lengths. We compressed the first 250 frames of each of the uncompressed video sequences using this variable GOP length encoder, creating a second database of unaltered videos. Frame deletion experiments run on these videos were used to simulate the aperiodic frame deletion fingerprints introduced by newer video compression techniques.

5.6.1 Frame Deletion Detection

To test the forensic effectiveness of our proposed frame deletion detectors, we first created a database of forged videos. To do this, we deleted 3, 6, and 9 frames from the beginning of each unaltered video sequence compressed using a fixed length GOP, then recompressed each video. This corresponded to removing 1/4, 1/2, and 3/4 of a GOP respectively. To test against frame addition, we added 6 frames to the beginning of each unaltered video sequence compressed with a fixed length GOP, then recompressed these videos. Additionally, we deleted 6 frames from the videos compressed using randomly varying GOP lengths. We then used each of our proposed detection techniques in conjunction with a series of different decision thresholds to determine if frame deletion or addition had occurred in each video.

The probabilities of detection P_d and false alarm P_{fa} were determined for each threshold by respectively calculating the percentage of forged videos that were correctly classified and the percentage of unaltered videos that were incorrectly classified. We used these results to generate the series of ROC curves for δ_{fixed} shown in Fig. 5.4 and for δ_{var} shown in Fig. 5.5. We can see from these ROC curves that both detectors' performance remains consistent regardless of the number of frames deleted. Furthermore, we can see that frame addition can be detected with the same accuracy as frame deletion. By examining the ROC curves for each detector corresponding to the average performance across all frame deletion amounts, we can see that both detectors were able to achieve at a P_d of at least 85% at a false alarm rate less than 5%. Both detectors also achieved a P_d of at least 90% at a false

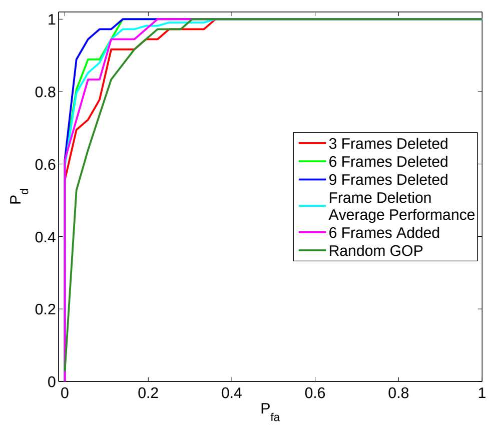


Figure 5.5: ROC curves for δ_{var} obtained by testing against different amounts frame deletion and addition.

alarm rate less than 10%. These results indicate that both detectors can be used to reliably detect frame deletion. Additionally, results presented in Fig. 5.5 suggest that δ_{var} can be used to detect frame deletion in videos compressed using randomly varying GOP lengths as well.

5.6.2 Frame Deletion Anti-Forensics

To evaluate the performance of our proposed frame deletion anti-forensic technique, we deleted six frames from each unaltered video compressed using a fixed GOP structure, then recompressed each video while applying our anti-forensic technique. When implementing our anti-forensic technique, we incorporated the modifications to our algorithm discussed in Section 5.4.

An example of typical results achieved by our proposed anti-forensic technique is shown in Fig. 5.6. This figure displays the P-frame prediction error sequence taken from an untampered MPEG compressed version of the 'Foreman' video, as well as

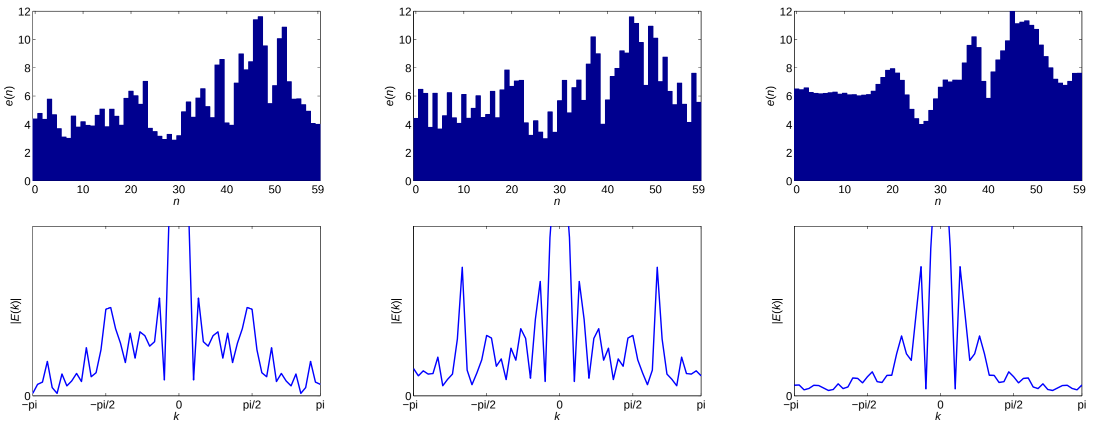
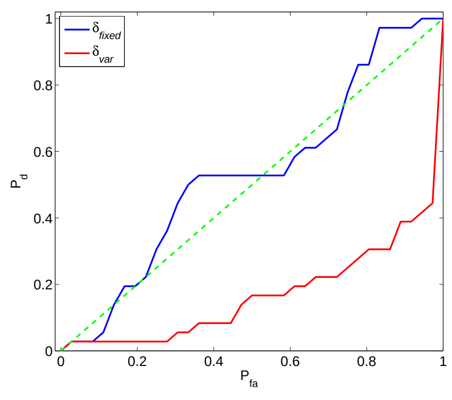


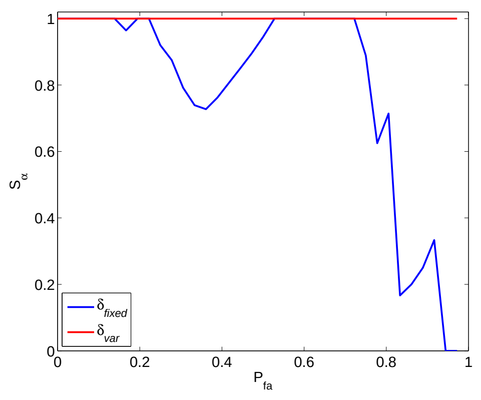
Figure 5.6: P-frame prediction error sequences (top row) and the magnitudes of their respective DFTs (bottom row) obtained from an untampered MPEG compressed version of the 'Foreman' video (left column), as well as from the same video after the first six frames were deleted followed by recompression without anti-forensic modification (middle column) and with the use of our proposed anti-forensic technique (right column).

the P-frame prediction error sequences obtained after deleting the first six frames and then recompressing the video with and without applying our anti-forensic temporal fingerprint removal technique. The temporal fingerprint features prominently in the prediction error sequence of the video in which frames are deleted without the use of our anti-forensic technique, particularly in the frequency domain. By contrast, these fingerprints are absent from the prediction error sequence when our anti-forensic technique is used to hide evidence of frame deletion.

Next, we examined the ability of our proposed anti-forensic technique to fool each of our automatic frame deletion detection techniques. To do this, we used both of our proposed detection techniques to classify each video in our databases of unaltered and anti-forensically modified MPEG-2 compressed videos as unaltered or one from which frames had been deleted. This was done using a series of different



(a)



(b)

Figure 5.7: Experimental results showing (a) ROC curves for δ_{fixed} and δ_{var} and (b) anti-forensic susceptibility plots for δ_{fixed} and δ_{var} obtained by testing on anti-forensically modified MPEG-2 videos.

decision thresholds; then the probabilities of detection and false alarm corresponding to each decision threshold were calculated from the results. We used this data to generate a new set of ROC curves for δ_{fixed} and δ_{var} when frame deletion has been disguised using anti-forensics. These ROC curves are displayed in Fig. 5.7(a).

In this figure, the dashed line represents the performance of a decision rule that randomly classifies a video as forged with a probability equal to P_{fa}. Reducing a detection technique's performance to this level corresponds to making it equivalent to a random guess. As we can see from Fig. 5.7(a), both frame deletion detection techniques perform at or near this level when our anti-forensic technique is applied to a video. Similarly, we used this data to compute the anti-forensic susceptibility of each detector to our proposed anti-forensic frame deletion technique. These results, which are displayed in Fig. 5.7(b), show that the detector δ_{var} is entirely susceptible to our anti-forensic technique at all false alarm levels. The detector δ_{fixed}

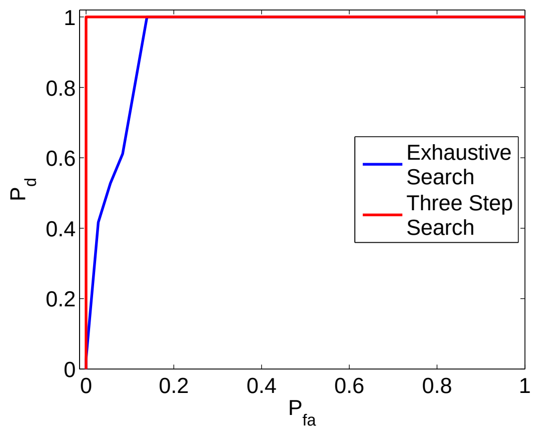


Figure 5.8: ROC curves for the anti-forensics detector δ_{mv} when tested on video data compressed using an exhaustive search to determine motion vectors and video data encoded using a three step motion vector search algorithm.

was slightly less susceptible to our anti-forensic attack, however, our anti-forensic technique achieved an anti-forensic susceptibility of .7 or greater for all $P_{fa} \leq 80\%$ for this detector. These results demonstrate that our proposed anti-forensic technique is able to render forensic frame deletion detection techniques nearly completely ineffective.

5.6.3 Detecting Frame Deletion Anti-Forensics

In order to evaluate the performance of our technique designed to detect the use of frame deletion anti-forensics, we re-examined the videos in our database of unaltered and anti-forensically modified MPEG-2 compressed videos. We used our proposed detector δ_{mv} to classify each video as unmodified or anti-forensically modified at a variety of different decision thresholds, then used these results to generate the ROC curve shown in Fig. 5.8.

The results of this experiment show that our proposed detector achieved perfect detection (i.e. a P_d of 100% at a P_{fa} of 0%). These results are slightly misleading, however, because the motion vectors of the videos in the unaltered database are obtained using an exhaustive search. Since an exhaustive search is also used when estimating a video's true motion during the detection of anti-forensics, there will be very little difference between an unaltered video's stored and recalculated motion vectors.

In reality, many video encoders use efficient algorithms to peform motion estimation. These algorithms greatly reduce the time needed to encode a video and produce a near optimal set of motion vectors. Nonetheless, the motion vectors obtained using these algorithms differ slightly from those obtained using an exhaustive search. As a result, it is more difficult to differentiate between an anti-forensically modified video and an unaltered video if one of these algorithms is used during encoding.

To evaluate the performance of our proposed frame deletion anti-forensics detection technique under less favorable conditions, we modified our video coder to perform motion estimation using the three step search algorithm proposed by Zhu and Ma [59]. We then created a new database of compressed unaltered videos whose motion vectors were obtained using this efficient search algorithm. We repeated the previous experiment with this data and used the results to generate the ROC curve shown in Fig. 5.8.

We can see from Fig. 5.8 that the performance of our proposed detector is degraded in this scenario. While the detection of frame deletion anti-forensics can still be performed, it must be done with a higher false alarm rate. This suggests that if a forensic investigator's maximum acceptable false alarm rate is sufficiently low, a video forger using anti-forensics is likely to avoid detection. To mitigate this, a forensic investigator may wish to repeat frame deletion anti-forensics detection using a decision threshold corresponding to a higher false alarm rate, but not immediately assume that detections correspond to forged videos. Instead, these videos can be flagged for closer investigation using additional forensic techniques.

5.6.4 Game Theoretic Evaluation of Video Forensics and Anti-Forensics

Once we evaluated the performance of each proposed forensic detection technique as well as the proposed video frame deletion anti-forensic technique, we used our game theoretic framework to identify the optimal strategies of both the forensic investigator and video forger. To do this, we modified our frame deletion anti-forensic technique to operate at variable strengths. This was accomplished by choosing the target P-frame prediction error sequence associated with strength k as

$$\tilde{e}_k(n) = k\tilde{e}(n) + (1-k)e(n), \tag{5.25}$$

where $\tilde{e}(n)$ denotes the fingerprint-free target prediction error sequence described in Section 5.3.

Because our proposed anti-forensic technique introduces virtually no distortion into a forged video, we set the term $\gamma(\cdot) = 0$ in the utility function of player 2. As a result, $U_2(k,\eta) = -U_1(k,\eta)$ causing our video forensic scenario to reduce to a zero

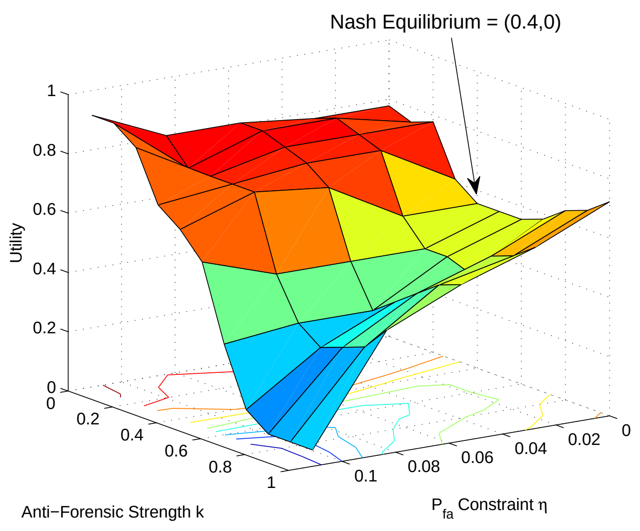


Figure 5.9: Utility function of the forensic investigator $U_1(k, \eta)$ when the total probability of false alarm constraint is $P_{fa}^{Tot} = 8.3\%$.

sum game. This allowed us to find the Nash equilibrium strategies by solving the following equation

$$(k^*, \eta^*) = \arg\max_{\eta} \min_{k} U_1(k, \eta). \tag{5.26}$$

Since no closed form expression for $U_1(k, \eta)$ exists in this scenario, we evaluated (5.26) numerically. This was done by first deleting frames from each single compressed video in our database, then anti-forensically modifying each video with strengths ranging between 0 and 1. For each anti-forensically modified video, we performed frame deletion detection and anti-forensics detection using a variety of different decision thresholds and then calculated the P_{fa} and P_d associated with each decision threshold and anti-forensic strength pairing. Using this data, the Nash equilibrium strategies and probability of forgery detection were calculated.

Fig. 5.9 shows the utility function $U_1(k, \eta)$ when the forensic investigator operates under the false alarm constraint $P_{fa}^{Tot} = 8.3\%$. Under this condition, the

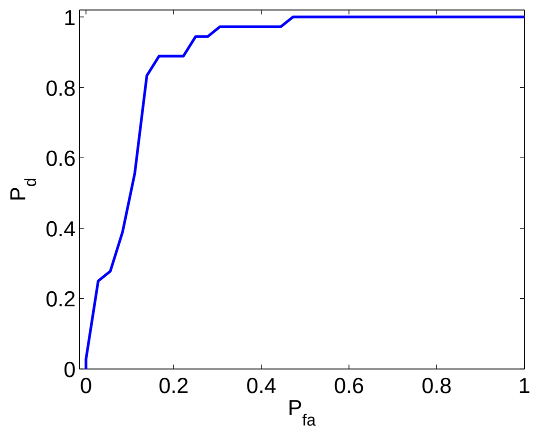


Figure 5.10: Nash equilibrium ROC curve for video frame deletion detection.

Nash equilibrium strategy is $k = 0.4$, $\eta = 0.0$, which corresponds to the forger reducing the strength of the anti-forensic attack to half and the forensic investigator allowing all of the false alarms to come from the anti-forensic detector δ_{mv}. The probability with which the forensic investigator will detect a forgery, i.e. the value of $U_1(k^*, \eta^*)$, is 38.9%. We note that it is less than the probability of detection achieved by both the frame deletion detector and the anti-forensics detector at the same false alarm level. This reinforces the notion that the forger can create a more successful anti-forensic attack by decreasing its strength.

We determined the Nash equilibrium strategies and calculated the probability of forgery detection for a set of total probability of false alarm constraints between 0% and 100%. We used these results to create the NE ROC curve displayed in Fig. 5.10. From this curve we can see that if the forensic investigator must operate with a total probability of false alarm constraint of 10% or less, frame deletion forgeries are difficult to detect. If the forensic examiner is able to relax their probability of false alarm constraint to roughly 15% or greater, frame deletion forgeries will be detected at a rate of at least 85%.

Table 5.1 shows the Nash equilibrium strategies for a variety of total probability of false alarm levels ξ. In some cases, multiple values of k are Nash equilibrium strategies for a particular value of ξ. We note that here, the value of U_1 corresponding to each Nash equilibrium strategy at a particular ξ value is the same. From the data presented in this table, we can observe two trends. The first is that as the false alarm constraint increases, the optimal strategy for the forger is to decrease the strength for applying anti-forensics. The second is that regardless of the value of ξ, the optimal strategy for the forensic investigator is to allow all of the false alarm contributions to come from the anti-forensics detector δ_{mv}. This is because the effectiveness of the frame deletion detection technique drops off quickly as k is increased. By contrast, the anti-forensics detector can still operate effectively even at low anti-forensic strengths. As a result, it is in the best interest of the forensic investigator to place the maximum load on δ_{mv}.

5.7 Summary

In this chapter, we have proposed a set of automatic frame deletion or addition detection techniques that operate by identifying increases in a video's P-frame prediction error that correspond to frame deletion or addition fingerprints. To do this, we first developed a model of a video's P-frame prediction error sequence before and after frame deletion or addition has occurred. Additionally, we used this model to design an anti-forensic technique capable of removing frame deletion or addition

Table 5.1: Nash equilibrium strategies k^* and η^* obtained for the forger and forensic investigator respectively at different constraints ξ on the forensic investigator's total probability of false alarm.

ξ	k^*	η^*
0.0%	0.6, 0.7, 0.8, 0.9, 1.0	*
2.8%	0.6	0.0%
5.6%	0.7	0.0%
8.3%	0.4	0.0%
11.1%	0.4, 0.5, 0.6	0.0%
13.9%	0.1, 0.2, 0.4	0.0%
16.7%	0.1, 0.2, 0.3, 0.5	0.0%
...	...	...

fingerprints form a forged video. This technique operates by first constructing a target prediction error sequence free from frame deletion or addition fingerprints, then modifying the motion vectors of each P-frame so that its total absolute prediction error matches the target value. Furthermore, we have proposed a forensic technique to detect the use of frame addition or deletion anti-forensics by comparing a compressed video's motion vectors to an estimate of the true motion in the video.

In addition to developing a set of forensic and anti-forensic techniques related to video frame deletion or addition, we have proposed a set of methods to evaluate the performance of anti-forensic techniques. These include an anti-forensic attack's probability of anti-forensic effectiveness and the anti-forensic susceptibility of a forensic detector to a particular anti-forensic attack. We have additionally proposed a game theoretic framework that can be used to understand the interplay between a forensic detector and a forger when the forger's use of anti-forensics can be detected. This framework allows us to identify the optimal set of actions for both the forensic investigator and forger, as well as to evaluate the ability of the forensic investigator to identify forgeries.

Through a series of simulations and experiments, we have evaluated the performance of each of our proposed forensic and anti-forensic techniques. Our results show that both of our proposed frame deletion or addition detection techniques can automatically detect video forgeries with a high degree of accuracy if anti-forensics is not used. These results also show that our proposed anti-forensic frame deletion or addition technique can successfully fool both forensic techniques. If this technique is applied at full strength, however, our anti-forensics detector is able to identify that anti-forensics has been used with a high degree of accuracy.

We have used our game theoretic framework to identify the optimal strategies for a forensic investigator and video forger to employ in a frame deletion or addition forgery scenario. These results show that as the forensic investigator's probability of false alarm constraint is increased, the strength with which the forger should apply anti-forensics is decreased. By contrast, the forensic investigator should allow the video frame addition or deletion detector to operate at a P_{fa} of 0% and allow all of the false alarms to come from the anti-forensics detector, regardless of the constraint on the total probability of false alarm. Furthermore, we have found that if the forensic investigator is bound by a total probability of false alarm constraint of approximately 10% or less, the forensic investigator will have less than a 50% chance of detecting a video forgery. If the total probability of false alarm constraint is above 15%, video forgeries can be detected at a rate of 85% or greater.

Chapter 6

Protection Against Reverse Engineering in Digital Cameras Using Anti-Forensics

Though the intended use of multimedia forensics is to provide information security, researchers have overlooked an important unintended use of forensic techniques: *multimedia forensics can be used to reverse engineer proprietary signal processing components in digital devices*! Digital cameras are an important example of this. Forensic techniques exist to estimate the color filter array (CFA) pattern and interpolation coefficients used during the image formation process [3, 6, 35, 53]. Furthermore, a camera's white balancing parameters can be forensically estimated [51]. Since camera manufacturers likely wish to protect their proprietary implementations of both color interpolation and white balancing, digital forensic techniques may in fact pose an intellectual property threat.

Because forensic techniques pose an information security threat when viewed in this light, *we propose using anti-forensics to protect against reverse engineering*. To accomplish this, we propose placing an anti-forensic processing module at the end of a device's internal signal processing pipeline. This will prevent forensic techniques from using a device's output to estimate signal processing operations inside the device.

In this chapter, we propose a proof-of-concept technique to prevent a digital camera's color interpolation method from being forensically reverse engineered. We accomplish this through a combination of nonlinear filtering and perturbations to an image's sampling grid. We demonstrate the effectiveness of our proposed technique by testing the ability of existing forensic algorithms to identify the color interpolation method used to form an image after our anti-forensic technique has been applied.

6.1 The Image processing pipeline

A digital camera operates by measuring the intensity of light reflected from a real world R scene onto an electronic sensor known as a charged coupling device (CCD), as is shown in Fig. 6.1. The light enters the camera by first passing through a lens. Since most CCDs are only capable of measuring one color of light at each pixel location, the light next passes through a color filter array P. The CFA is an optical filter consisting of a repeating fixed pattern (typically 2x2) which allows only one color band of light (red, green, or blue) to fall incident on the CCD at a particular pixel location.

The CCD then measures the light intensity of the corresponding color band at each pixel location. This yields an image S constructed of three partially sampled color layers such that

$$S_{x,y,c} = \begin{cases} R_{x,y,c} & \text{if } P_{x,y} = c, \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

where x and y are indices denoting a pixel's spatial location and c specifies its color layer.

Next, unobserved color layer values at each pixel location are interpolated

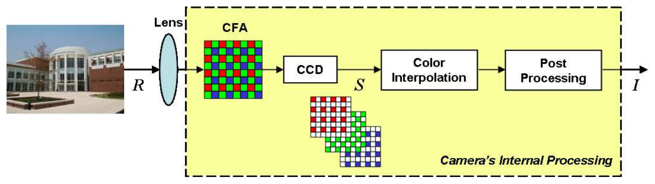


Figure 6.1: A digital camera's signal processing pipeline.

using nearby directly observed color layer values. This interpolation process can be performed in many ways and is typically camera model specific. After this, the image may be subject to internal post-processing, such as white balancing, before it is stored or output.

6.1.1 Component Forensics

Knowledge of a camera's color interpolation coefficients and CFA pattern can be used to perform a variety of forensic tasks. Forensic techniques that estimate a camera's color interpolation coefficients and CFA pattern, then use these to perform another forensic task, are known as *component forensic* techniques. Component forensic techniques have been developed to identify forgeries by detecting localized interpolation irregularities [35, 52]. Because interpolation methods and their parameters are typically camera model specific, other component forensic techniques have been developed to identify an image's source camera [3, 6, 53]. Others use knowledge of the CFA pattern and interpolation coefficients to estimate parameters of a camera's internal post-processing operations such as white balancing [51].

While component forensic techniques vary in the specific way that they estimate a camera's color interpolation coefficients, they all share the same basic structure. Here we use the technique proposed by Swaminathan et al. [53] as a baseline and describe how it operates.

First, an image's CFA pattern is assumed. By doing this, a forensic examiner can separate directly observed pixels in a color layer from those that have been interpolated. Next, the directly observed color values are used to calculate the horizontal and vertical gradients of each pixel. These are used to classify each each pixel into one of three sets for each color layer depending on the strength of its horizontal and vertical gradient. For each of the nine pairings of color layer and gradient class, the directly observed and interpolated color layer values are used to obtain a least squares estimate of the color interpolation filter coefficients.

Since in most cases the true CFA pattern is not known, this process is repeated for each of the 36 possible 2×2 CFA patterns. After the set of interpolation coefficients is estimated for a candidate CFA pattern, each color layer is resampled using the candidate CFA and the color layers are interpolated using the estimated coefficients. The difference between the original image and the re-interpolated image is then calculated for each set of estimated interpolation coefficients and CFA pattern. The CFA pattern and interpolation coefficients that result in the lowest difference are chosen as the final estimate.

The estimated color interpolation coefficients can be used to train a support vector machine (SVM) to identify the color interpolation method used or identify the model of the camera used to capture an image.

6.2 Anti-Forensic Reverse Engineering Prevention

A party attempting to reverse engineer a digital camera will likely try to determine the color interpolation method used by that camera and estimate its color interpolation coefficients. Though reverse engineering is not the intended use of component forensic techniques, they can be used by a reverse engineer to accomplish this. As a result, camera manufacturers may wish to incorporate some form of protection against reverse engineering into their devices.

Swaminathan et al.'s technique obtains an estimate of the camera's color interpolation coefficients as follows. For a given color layer and CFA pattern, each interpolated pixel b is written as a linear combination of nearby directly observed pixel values a according to the equation

$$b_{x,y} = \sum_{(i,j)\in\Omega_I} w_{i,j}^{(I)} a_{x+i,y+j} \tag{6.2}$$

where $w^{(I)}$ is the interpolation filter and Ω_I is its support. These equations are grouped by color layer and gradient class into systems of equations of the form $\mathbf{Aw} = \mathbf{b}$. A least squares approximation of the interpolation filter coefficients is calculated for each set of equations, resulting in nine sets of filter coefficients.

To combat forensic reverse engineering, we propose incorporating an anti-forensic module into a digital camera's processing pipeline as is shown in Fig. 6.2. This module is designed to interfere with two important aspects of component forensic algorithms:

1. The estimate of the interpolation method is linear.

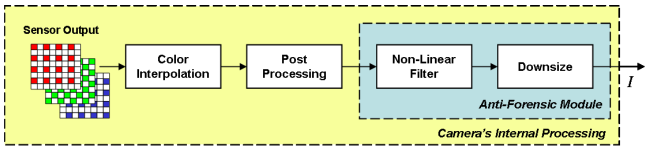


Figure 6.2: A digital camera's internal processing pipeline with our proposed anti-forensic module integrated into it.

2. This linear estimate depends on the ability of the forensic algorithm to guess which color layer values were directly observed and which were interpolated.

The first element of our anti-forensic module is a nonlinear filter. This is used to reduce linear dependencies between interpolated pixel values and nearby directly observed pixel values. In this proof-of-concept implementation, we use a 2×2 median filter to perform nonlinear filtering. Letting d denote an input pixel value and f denote an output pixel value, the 2×2 median filter is defined as

$$f_{x,y} = \text{median}\{d_{x,y}, d_{x+1,y}, d_{x,y+1}, d_{x+1,y+1}\}. \tag{6.3}$$

The second element of our anti-forensic module involves downsizing the image by a small factor. This is done to disrupt the color sampling grid and prevent the forensic algorithm from identifying directly observed and interpolated color values. Each pixel in the downsized image will correspond to a greater effective area than in the originally sized image. As a result, no pixel in the downsized image will correspond solely to a directly observed or color interpolated pixel. This phenomenon is shown in Fig. 6.3.

In this proof-of-concept implementation, we downscale using bilinear interpolation. Let each color layer be $X \times Y$ pixels before downsizing and $P \times Q$ pixels after. Also, let the integer pixel location (p, q) in the downsized image corresponds the real valued location (u, v) in the originally sized image. These locations are related according to the equations $u = (p(X-1) + P - X)/(P-1)$ and $v = (q(Y-1) + Q - Y)/(Q-1)$. Additionally, let $x \leq u < x+1$ and $y \leq v < y+1$ as is shown in Fig. 6.4. Each pixel g in the downscaled color layer is given by

$$g_{u,v} = \sum_{(k,l)\in\Omega_D} w_{k,l}^{(D)}(u,v) f_{x+k,y+l}, \tag{6.4}$$

where $\Omega_D = \{(0,0),(0,1),(1,0),(1,1)\}$ and $w_{k,l}^{(D)}(u,v)$ are the spatially varying downscaling coefficients. The coefficients of the bilinear downscaling filter are calculated using the following equations: $w_{0,0}^{(D)}(u,v) = (1-u+x)(1-v+y)$, $w_{0,1}^{(D)}(u,v) = (1-u+x)(v-y)$, $w_{1,0}^{(D)}(u,v) = (u-x)(1-v+y)$, and $w_{1,1}^{(D)}(u,v) = (u-x)(v-y)$.

Combining (6.3), (6.4), and the expressions relating p and q to u and v, the output of the anti-forensic module can be written as

$$\begin{aligned} g_{p,q} = \sum_{(k,l)\in\Omega_D} \Big(& w_{k,l}^{(D)}\big(\tfrac{p(X-1)+P-X}{P-1}, \tfrac{q(Y-1)+Q-Y}{Q-1}\big) \\ & \times \operatorname{median}\{d_{x+k,y+l}, d_{x+k+1,y+l}, d_{x+k,y+l+1}, d_{x+k+1,y+l+1}\}\Big). \end{aligned} \tag{6.5}$$

When our proposed anti-forensic module is employed, both directly observed and interpolated color layer values are modified according to this expression. As a result, both a and b are modified in (6.2) causing the least squares estimate to result in a poor approximation of the interpolation method.

In practice we have found that the image needs to be downscaled only a minimal amount in order to protect against forensic reverse engineering. We explore this in further detail in Section 6.3.

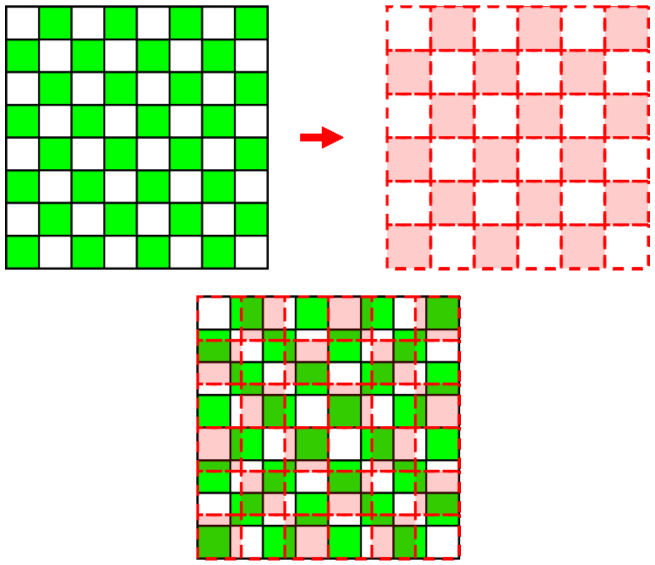


Figure 6.3: Top: Changes in the effective area of each pixel after downsizing. Bottom: A downsized color layer overlaid on the pixels of the original color layer.

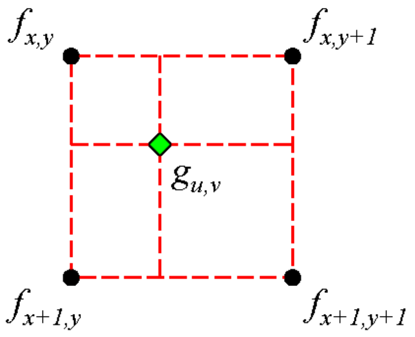


Figure 6.4: Bilinear interpolation example.

6.3 Simulations and Results

6.3.1 Performance Metric

In order to assess our proposed anti-forensic technique's ability to protect against reverse engineering, we first need to establish a measure of the attacking component forensic technique's performance.

Let M and $\hat{M}$ be random variables representing the color interpolation method used to form an image and the method identified using component forensics respectively. Additionally, let $\mathcal{M}$ be the set of all candidate interpolation methods. For a given interpolation method $m \in \mathcal{M}$, the probability $P_C^{(m)}$ that interpolation method m was correctly identified by the component forensic technique is $P_C^{(m)} = P(M = m|\hat{M} = m)$. We measure the performance of the component forensic technique by evaluating this probability for all $m \in \mathcal{M}$.

We note that if $P_C^{(m)} \leq 1/|\mathcal{M}|$, where $|\mathcal{M}|$ denotes the cardinality of the set $\mathcal{M}$, then the component forensic technique performs no better than a random guess. Ideally, we would like our anti-forensic module to reduce $P_C^{(m)}$ below $1/|\mathcal{M}|$ for all $m \in \mathcal{M}$, thus rendering the component forensic technique's output completely unreliable.

6.3.2 Experimental Results

We created a test database of images whose color interpolation method and CFA pattern was known as ground truth in order to experimentally evaluate the performance of our proposed anti-forensic reverse engineering prevention technique.

This was done by first creating a set of 100 272×272 pixel images cropped from the center of images in the Uncompressed Colour Image Database [38]. Next, we resampled the color components of each image using the Bayer pattern as the CFA pattern. We then performed color interpolation using five different color interpolation methods: bilinear, bicubic, nearest neighbor, median filter, and smooth hue transition. Descriptions of the interpolation methods used can be found in [53] and [35]. (We note that the median filter color interpolation technique is not simply applying a median filter to each color layer.)

The resulting 500 images model the direct output of a digital camera. Because post-processing such as compression decreases the performance of component forensic techniques [53], we did not subject these images to post-processing. This allowed us to evaluate the performance of our anti-forensic module operating under worst case conditions; i.e. ideal conditions for component forensic techniques. Additionally, this mimics the settings that would likely be chosen by someone wishing to reverse engineer a camera using component forensics with access to the camera. Finally, we passed these images through our proposed anti-forensic module. We varied the downscaling amount between 0 and 28 pixels in 4 pixel increments. This created 4000 anti-forensically modified images in addition to the 500 unmodified images. Fig. 6.5 shows an example of an image before and after it passes through our anti-forensic module.

After constructing our image database, we used the component forensic technique proposed by Swaminathan et al. in [53] to estimate the CFA pattern and color interpolation coefficients for each of the 4500 images. We then trained a sup-

Figure 6.5: Left: A typical image formed using bilinear color interpolation. Right: The same image after being passed through our anti-forensic module.

port vector machine (SVM) with a radial basis function kernel to identify the color interpolation method used to form each image [4].

To achieve a baseline assessment of the component forensic technique's ability to identify each color interpolation method, we first evaluated it using only unmodified images. This was done using cross validation by dividing the set of unmodified images into 10 subsets. The color interpolation method was identified for every image in a given subset after training the SVM using the remaining 9 subsets. This process was repeated for each of the 10 subsets. The results were used to calculate $P_C^{(m)}$ for each interpolation method according to the equation

$$P_C^{(m)} = \sum_n \frac{\mathbb{1}(M_n = m, \hat{M}_n = m)}{\mathbb{1}(\hat{M}_n = m)}. \tag{6.6}$$

where n is the picture index and $\mathbb{1}(\cdot)$ is the indicator function. When testing on unmodified images, the component forensic technique achieved perfect performance, i.e. $P_C^{(m)} = 1$ for each of the 5 color interpolation techniques.

Next, we tested the effectiveness of our anti-forensic module. We did this by using the trained SVM to identify the color interpolation method used to form each of the 4000 anti-forensically modified images. To ensure that image content had

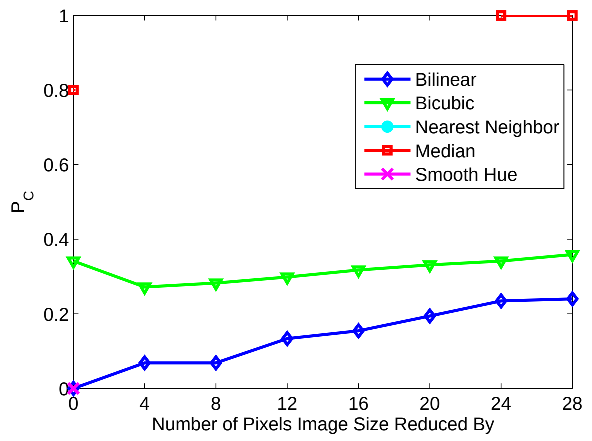


Figure 6.6: Plot of downscaling amount vs. P_C.

no influence on the identification results, the anti-forensically modified images were divided into 10 subsets corresponding to the unmodified training images. During testing, the SVM was trained using the 9 subsets of training data corresponding to the unused testing subsets. This data was used to calculate $P_C^{(m)}$ for every pairing of interpolation method and downscaling amount.

These results of this test are shown in Fig. 6.6. We note that in this figure, $P_C^{(m)}$ is not displayed for some interpolation techniques at certain downscaling amounts. This is because for these downscaling amounts, no images were identified as being formed using these interpolation techniques. As a result, $P_C^{(m)}$ is undefined in these situations. This indicates ideal perfomance for the anti-forensic module, since the component forensic algorithm was unable to correctly identify these interpolation techniques.

We find that our proposed anti-forensic module achieves its best performance when images are downsized by only 4 pixels. This is advantageous because downsizing by such a minor amount minimizes undesired effects caused by downsizing.

Since 5 candidate interpolation methods were considered, the component forensic technique performs better than a random guess only when $P_C^{(m)} \leq 0.2$. When our anti-forensic module downsizes the image by 4 pixels, our results show that $P_C^{(m)} > 0.2$ only when bicubic color interpolation is considered. Even in this case, $P_C^{(bicubic)} = 0.27$ which exhibits little improvement over a random guess. These results suggest that our proposed anti-forensic module is very effective at protecting against reverse engineering.

The use of our anti-forensic module, particularly the nonlinear filter component, will introduce some distortion into the output image. To mitigate this, we can chose not to apply median filtering to ever pixel in the image. Instead, we can randomly select a number of pixels to filter and leave the rest unaltered. This comes at a cost, however, in the form of an increase in $P_C^{(m)}$. Depending on the image quality constraints of the camera manufacturer and the desired level of reverse engineering protection, a manufacturer can choose a balance between these quantities that best meets its needs.

To characterize the trade-off between image quality and reverse engineering protection, we varied the probability that a pixel would be median filtered during anti-forensic modification between 0% and 100%, then repeated the previous experiment. In this test, each image was downscaled by 4 pixels in each direction. We then calculated $P_C^{(m)}$ for each interpolation method at each pixelwise median filtering probability. Additionally, we measured the distortion introduced by our anti-forensic module by measuring the average structual similarity (SSIM) between each image before and after anti-forensic modification [56]. The results of this ex-

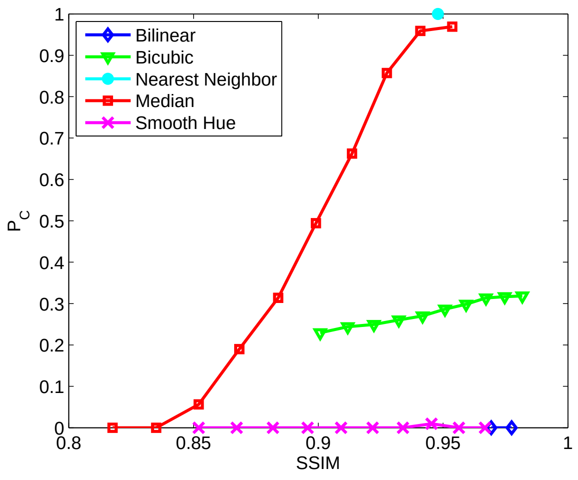


Figure 6.7: Plot of SSIM vs. P_C.

periment are shown in Fig. 6.7. From this figure we can see that $P_C^{(m)}$ can either be kept equivalent to a random guess or held to its minimum value while achieving a SSIM of .85 or greater. We note that for nearest neighbor color interpolation, a SSIM of greater than .92 was achieved without the component forensic algorithm being able to identify any images as having been nearest neighbor interpolated (thus making $P_C^{(m)}$ undefined).

6.4 Summary

In this chapter, we have proposed a new anti-forensic module to be incorporated into a digital camera's signal processing pipeline to protect against reverse engineering. By introducing nonlinearities into an image and disrupting its color sampling grid, our anti-forensic module prevents component forensic techniques from accurately estimating the color interpolation method used by a digital camera during the image formation process. Through a set of experiments, we have demonstrated that our proposed anti-forensic technique is able to reduce the performance of a component forensic technique to that of a random guess or worse in nearly all cases test.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this dissertation, we have examined the problem of authenticating multimedia signals using digital forensic techniques. By identifying the fingerprints of several editing operations, we have have developed new techniques to identify edited and falsified digital multimedia content. Additionally, we have taken the novel step of examining multimedia security from the forger's point of view. We have shown that a forger can design anti-forensic operations and use them to hide evidence of file manipulation. By studying both forensics and anti-forensics, we hope to provide a more complete view of multimedia information security.

In Chapter 2, we identified the fingerprints left in a digital image's pixel value histogram by pixel value mappings. Using these fingerprints, we proposed a forensic technique to detect the use of contrast enhancement on digital images. We identified the specific fingerprints of histogram equalization, and developed a detector to determine if this specific contrast enhancement operation was used. Since a forger often must use contrast enhancement to ensure that lighting conditions match when creating a cut-and-paste forgery, we showed that cut-and-paste forgeries can be identified by performing localized contrast enhancement detection. Additionally, we proposed a technique to detect the addition of noise to a previously JPEG compressed image by showing that additive noise prevents the fingerprints of a specific pixel value mapping from arising.

In Chapter 3, we developed a method to jointly estimate the contrast enhancement mapping used to modify a digital image as well as the image's pixel value histogram before it was contrast enhanced. To do this, we developed a probabilistic model of an image's pixel value histogram. We used this model to identify the histogram entries that were most likely to correspond to contrast enhancement fingerprints, then estimated the contrast enhancement mapping used to create these fingerprints.

In Chapter 4, we proposed a set of anti-forensic techniques capable of removing compression fingerprints from a digital image. To remove the transform coefficient quantization fingerprints that arise from image compression, we developed a framework that operates by first modeling the image's unaltered transform coefficient distribution, then adding anti-forensic dither to the quantized transform coefficients. We proved that our choice of anti-forensic dither distributions will completely remove all quantization artifacts and result in an anti-forensically modified transform coefficient distribution that matches the unaltered image's transform coefficient distribution. Additionally, we proposed a technique capable of removing statistical traces of blocking artifacts that commonly arise during JPEG compression. We demonstrated that these techniques can be used to create undetectable forgeries and falsify an image's origin.

In Chapter 5, we examined the problem of detecting frame deletion in motion compensated digital video. We developed a set of forensic techniques that are capable of detecting frame deletion when a video encoder uses a fixed or variable length group of picture sequence. We proposed an anti-forensic technique capable of hiding frame deletion fingerprints and showed that it can be used to hide evidence of video frame deletion. Furthermore, we developed a method to detect the use of frame deletion anti-forensics.

Because a forgery can be detected by identifying editing fingerprints or fingerprints left by anti-forensics, we showed that a forger must balance the strength with which anti-forensics are applied. Similarly, we showed that since a forensic investigator must employ both an editing detector and an anti-forensics detector, a forensic investigator must find the optimal balance between the false alarm rates of both detectors. We showed that both a forger and forensic investigator's optimal choice of actions are dependent on the actions of the other party and proposed a game theoretic framework to identify the set of actions which neither party has an incentive to deviate. We applied our game theoretic framework to the video frame deletion forensic scenario and identified under which conditions a video forgery is likely to be detected.

In Chapter 6, we demonstrated that anti-forensics can be used to prevent reverse engineering in digital devices. To prevent component forensic techniques from estimating a digital camera's color interpolation algorithm, we proposed incorporating an anti-forensic module consisting of nonlinear filtering and perturbations to an image's sampling grid into digital cameras. We showed that this technique prevents component forensic techniques from identifying a digital camera's interpolation method at a rate slightly better than a random guess.

7.2 Future Work

A major unexamined challenge arises from the fact that in practice, multiple forensic tests must be run to identify the many ways that a multimedia file may be altered. Currently, little research exists describing how to optimally combine the results of multiple forensic tests. I plan to develop theoretical techniques to both merge the results of several forensic tests and control the total false alarm rate when multiple operation-specific forensic tests are used. To do this, I will examine the effect of a detection technique's accuracy and the probability that a forgery operation is used on the optimal weight of a forensic test.

As I have shown in my anti-forensic research, there are many ways an intelligent forger can attempt to avoid detection. I am interested in researching new methods that a forger can employ to avoid forgery detection and developing new forensic techniques in response. For example, a forger may take advantage of the variability in the accuracy of different forensic tests by choosing to only use editing operations that are difficult to forensically identify. As a result, a forensic investigator must adjust the detection strategy to compensate for this change in a forger's behavior. I will use game theory and other techniques to study the dynamic interplay between an intelligent forger and a forensic investigator. Based on my findings, I will determine how to optimally detect multimedia forgeries when dealing with an intelligent forger who actively evades forensic detection.

Currently, most forensic techniques are designed to detect the use of an individual editing operation or identify inconsistencies in a specific property of a multimedia signal. In reality, however, many editing operations can and are used to create a forgery. I am interested in addressing the many research issues that arise due to this disconnect. For example, forensic techniques that detect a specific editing operation often work by identifying unique traces left by that operation in a multimedia signal. Subsequent editing and processing may alter these traces and affect the accuracy of forensic techniques. I plan to study the interactions between different multimedia processing operations and the effects that they have on forensic traces. I will use this research to improve the robustness of forensic techniques, and tackle new problems such as identifying the order in which editing operations were applied. This may provide insight into the forgery creation process and help identify who performed malicious tampering if a multimedia file was processed by several users.

Bibliography

[1] I. Avcibaş, S. Bayram, N. Memon, M. Ramkumar, and B. Sankur. A classifier design for detecting image manipulations. In *Proc. IEEE Int. Conference on Image Processing*, volume 4, pages 2645–2648, October 2004.

[2] S. Bayram, I. Avcibaş, B. Sankur, and N. Memon. Image manipulation detection. *Journal of Electronic Imaging*, 15(4):041102, 2006.

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

[4] C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

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

[6] W. H. Chuang and M. Wu. Semi non-intrusive training for cell-phone camera model linkage. In *Proc. IEEE Workshop on Information Forensics and Security*, pages 1 –6, Seattle, WA, December 2010.

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

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

[9] H. Farid. Digital image ballistics from JPEG quantization. Technical Report TR2006-583, Dept. of Computer Science, Dartmouth College, 2006.

[10] J. Fridrich. Image watermarking for tamper detection. In *IEEE Int. Conference on Image Processing*, volume 2, pages 404 –408 vol.2, Chicago, IL, USA, October 1998.

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

[12] T. Gloe, M. Kirchner, A. Winkler, and R. Böhme. Can we trust digital image forensics? In *Proc. 15th Int. Conference on Multimedia*, pages 78–86, 2007.

[13] M. Goljan, J. Fridrich, and M. Chen. Defending against fingerprint-copy attack in sensor-based camera identification. *IEEE Trans. on Information Forensics and Security*, 6(1):227 –236, March 2011.

[14] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[15] Calvin Haas. JPEGsnoop - JPEG file decoding utility. `http://www.impulseadventure.com/photo/jpeg-snoop.html`.

[16] J. He, Z. Lin, L. Wang, and X. Tang. Detecting doctored JPEG images via DCT coefficient analysis. In *Proc. European Conference on Computer Vision*, volume 3593, pages 423–435, May 2006.

[17] G. E. Healey and R. Kondepudy. Radiometric CCD camera calibration and noise estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(3):267–276, March 1994.

[18] M. K. Johnson and H. Farid. Exposing digital forgeries by detecting inconsistencies in lighting. In *Proc. ACM Multimedia and Security Workshop*, pages 1–10, New York, NY, USA, 2005.

[19] M. K. Johnson and H. Farid. Exposing digital forgeries through chromatic aberration. In *Proc. ACM Multimedia and Security Workshop*, pages 48–55, Geneva, Switzerland, 2006.

[20] M. K. Johnson and H. Farid. Exposing digital forgeries in complex lighting environments. *IEEE Trans. on Information Forensics and Security*, 2(3):450–461, September 2007.

[21] M. Kirchner and R. Böhme. Hiding traces of resampling in digital images. *IEEE Trans. on Information Forensics and Security*, 3(4):582–592, December 2008.

[22] M. Kirchner and R. Böhme. Synthesis of color filter array pattern in digital images. In *Proc. SPIE-IS&T Electronic Imaging: Media Forensics and Security*, volume 7254, February 2009.

[23] E. Y. Lam and J. W. Goodman. A mathematical analysis of the DCT coefficient distributions for images. *IEEE Trans. on Image Processing*, 9(10):1661–1666, October 2000.

[24] J. Li and R. M. Gray. Text and picture segmentation by the distribution analysis of wavelet coeffcients. In *Proc. IEEE Int. Conference on Image Processing*, pages 790 – 794, October 1998.

[25] A. W. C. Liew and H. Yan. Blocking artifacts suppression in block-coded images using overcomplete wavelet representation. *IEEE Trans. on Circuits Systems for Video Technology*, 14(4):450–461, April 2004.

[26] W. S. Lin, S. K. Tjoa, H. V. Zhao, and K. J. Ray Liu. Digital image source coder forensics via intrinsic fingerprints. *IEEE Trans. on Information Forensics and Security*, 4(3):460–475, September 2009.

[27] J. Lukáš, J. Fridrich, and M. Goljan. Detecting digital image forgeries using sensor pattern noise. In *Proc. SPIE, Electronic Imaging, Security, Steganography, Watermarking of Multimedia Contents*, volume 6072, pages 362–372, San Jose, CA, USA, February 2006.

[28] J. Lukáš and J. Fridrich. Estimation of primary quantization matrix in double compressed JPEG images. pages 5–8, August 2003.

[29] T. T. Ng, S. F. Chang, J. Hsu, L. Xie, and M.P. Tsui. Physics-motivated features for distinguishing photographic images and computer graphics. In *Proc. ACM Multimdedia*, pages 239–248, Singapore, 2005.

[30] T. T. Ng, S. F. Chang, and Q. Sun. Blind detection of photomontage using higher order statistics. In *Proc. IEEE Int. Symp. Circuits Systems*, volume 5, pages V–688–V–691, Vancouver, BC, Canada, May 2004.

[31] M. Nizza and P. J. Lyons. In an Iranian image, a missile too many. New York Times News Blog, July 2008. `http://thelede.blogs.nytimes.com/2008/07/10/in-an-iranian-image-a-missile-too-many/`.

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

[33] A. C. Popescu and H. Farid. Statistical tools for digital forensics. In *Proc. 6th Int. Workshop Information Hiding*, pages 128–147, Toronto, Canada, 2004.

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

[35] A. C. Popescu and H. Farid. Exposing digital forgeries in color filter array interpolated images. *IEEE Trans. on Signal Processing*, 53(10):3948–3959, October 2005.

[36] J. R. Price and M. Rabbani. Biased reconstruction for JPEG decoding. *IEEE Signal Processing Letters*, 6(12):297–299, December 1999.

[37] A. Said and W.A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Technology*, 6:243–250, June 1996.

[38] G. Schaefer and M. Stich. UCID: an uncompressed color image database. In *Proc. SPIE: Storage and Retrieval Methods and Applications for Multimedia*, volume 5307, pages 472–480, 2003.

[39] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36 –58, September 2001.

[40] M. Stamm and K.J.R. Liu. Blind forensics of contrast enhancement in digital images. In *Intl. Conference on Image Processing*, pages 3112–3115, Oct. 2008.

[41] M. C. Stamm, W. S. Lin, and K. J. R. Liu. Forensics vs. anti-forensics: A decision and game theoretic framework. In *Proc. IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, pages 1749–1752, Kyoto, Japan, March 2012.

[42] M. C. Stamm, W. S. Lin, and K. J. R. Liu. Temporal forensics and anti-forensics in digital videos. *to appear in IEEE Trans. on Information Forensics and Security*, 2012.

[43] M. C. Stamm and K. J. R. Liu. Forensic detection of image tampering using intrinsic statistical fingerprints in histograms. In *Proc. APSIPA Annual Summit and Conference*, October 2009.

[44] M. C. Stamm and K. J. R. Liu. Forensic detection of image manipulation using statistical intrinsic fingerprints. *IEEE Trans. on Information Forensics and Security*, 5(3):492 –506, September 2010.

[45] M. C. Stamm and K. J. R. Liu. Forensic estimation and reconstruction of a contrast enhancement mapping. In *Proc. IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, pages 1698 – 1701, March 2010.

[46] M. C. Stamm and K. J. R. Liu. Wavelet-based image compression anti-forensics. In *Proc. IEEE Int. Conference on Image Processing*, pages 1737 – 1740, September 2010.

[47] M. C. Stamm and K. J. R. Liu. Anti-forensics for frame deletion/addition in MPEG video. In *Proc. IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, pages 1876 – 1879, Prague, Czech Republic, May 2011.

[48] M. C. Stamm and K. J. R. Liu. Anti-forensics of digital image compression. *IEEE Trans. on Information Forensics and Security*, 6(3):1050 –1065, September 2011.

[49] M. C. Stamm, S. K. Tjoa, W. S. Lin, and K. J. R. Liu. Anti-forensics of JPEG compression. In *Proc. IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, pages 1694 – 1697, March 2010.

[50] M. C. Stamm, S. K. Tjoa, W. S. Lin, and K. J. R. Liu. Undetectable image tampering through JPEG compression anti-forensics. In *Proc. IEEE Int. Conference on Image Processing*, pages 2109 – 2112, September 2010.

[51] A. Swaminathan, M. Wu, and K.J.R. Liu. Optimization of input pattern for semi non-intrusive component forensics of digital cameras. In *Proc. IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, pages II–225 –II–228, Honolulu, HI, April 2007.

[52] A. Swaminathan, M. Wu, and K.J.R. Liu. Digital image forensics via intrinsic fingerprints. *IEEE Trans. on Inform. Forensics Security*, 3(1):101–117, March 2008.

[53] A. Swaminathan, Min Wu, and K.J.R. Liu. Nonintrusive component forensics of visual sensors using output images. *IEEE Trans. on Information Forensics and Security*, 2(1):91 –106, March 2007.

[54] G. Valenzise, V. Nobile, M. Taglisacchi, and S. Tubaro. Countering JPEG anti-forensics. In *Proc. IEEE Int. Conference on Image Processing*, Brussels, Belgium, September 2011.

[55] W. Wang and H. Farid. Exposing digital forgeries in video by detecting double MPEG compression. In *Proc. ACM Multimedia and Security Workshop*, pages 37–47, Geneva, Switzerland, 2006.

[56] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing*, 13(4):600 –612, April 2004.

[57] S. Ye, Q.n Sun, and E.-C. Chang. Detecting digital image forgeries by measuring inconsistencies of blocking artifact. In *Proc. IEEE Int. Conference on Multimedia Expo*, pages 12–15, 2007.

[58] G. Zhai, W. Zhang, X. Yang, W. Lin, and Y. Xu. Efficient image deblocking based on postfiltering in shifted windows. *IEEE Trans. on Circuits Systems Video Technology*, 18(1):122–126, January 2008.

[59] S. Zhu and K. K. Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Trans. on Image Processing*, 9:287–290, February 2000.