SECURE AND ROBUST COMPRESSED-DOMAIN VIDEO WATERMARKING FOR H.264

A Thesis
Presented to
The Academic Faculty

by

Maneli Noorkami

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2007

SECURE AND ROBUST COMPRESSED-DOMAIN VIDEO WATERMARKING FOR H.264

Approved by:

Dr. Russell M. Mersereau,
Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Steven W. McLaughlin
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Aaron D. Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Nikil S. Jayant
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Jarek Rossignac
College of Computing
Georgia Institute of Technology

Date Approved: 21 May, 2007

This dissertation is dedicated to

my parents, Pari and Morteza, and to my sister, Paria.

Thank you for your love, encouragement and support.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor, Dr. Russell Mersereau, for being an excellent advisor, and for his support, inspiration and kindness. In the many years that I spent in school, he is the best professor I ever had and I feel very fortunate to have him as my Ph.D. advisor. Through all the meetings I had with him during the last four years, I have always left his office happier, more encouraged and motivated and I am very thankful for that. He is not only a great advisor, but a wonderful person as well. Working with him has made this journey very pleasant, and for that, I am deeply grateful.

I would like to thank my committee members, Dr. Steven McLaughlin, Dr. Aaron Lanterman, Dr. Nikil Jayant, and Dr. Jarek Rossignac for their time and valuable suggestions. I would especially like to thank Dr. Aaron Lanterman for his insightful discussions on detection theory.

I would like to thank Texas Instruments for partially supporting this research. I would also like to thank Bruce Flinchbaugh for giving me the opportunity to spend two internships at Texas Instruments. My special thanks goes to Minhua Zhou at Texas Instruments for his suggestion to work on the new video coding standard H.264. His expertise in video coding has been extremely helpful several times in my research, and I am very grateful that I met him.

I would like to extend my gratitude to Dr. Ghassan AlRegib for sharing his experiences with me from when he was a graduate student at Georgia Tech. I also would like to thank Dr. Faramarz Fekri for giving me the opportunity to come to Georgia Tech and for teaching me the first steps in conducting academic research.

I would like to thank the staff at the Center for Signal and Image Processing

(CSIP) for taking care of the daily business of the lab. I would especially like to thank Keith May for his help with the computer facilities. My best regards also go to my peers, faculty and staff at CSIP.

I am thankful to my close friend Mina Sartipi, for being a great roommate in the first few years of my stay at Georgia Tech. We have experienced a great deal together. I am also thankful to her for discussing my research when needed. I also would like to thank my close friend Will Leven for proofreading the first draft of my work and for being a great listener when I needed to talk about my research problems. Mina Raiszadeh has been my best friend since she came to Georgia Tech, I am thankful to her for her friendship. I would also like to give many thanks to my other friends, David Boivin, Kevin Chan, Nicolas Gastaud, Marjan Mehdizadeh, Nusha Safabakhsh, Azin Sahabi, Neda Sedighi, Jerome Vasseur, Pegah Zamani and many others who have brought joy and laughter into my life. I am thankful to Rajbabu Velmurugan for being a great neighbor all these years. I would like to thank Antonio Moreno for his help with clusters and for writing a script that expedited running my simulations considerably.

Last but not least, my love and gratitude go to my parents, Pari and Morteza, for raising me with their ideal of always striving for excellence, for valuing my education and for putting my success and happiness the first priority in their lives. I am also grateful for their unconditional love and support at a great distance. I am thankful to my sister, Paria, for advising me and showing me the path to success, and for her patience with her younger sister. Her love and encouragement has always been a constant source of comfort. I would also like to thank by brother-in-law, Mehran Azimi, being in the same field, his advice and help has been very valuable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of this thesis is to present a robust watermarking algorithm for H.264 and to address challenges in compressed-domain video watermarking. To embed a perceptually invisible watermark in highly compressed H.264 video, we use a human visual model. We extend Watson's human visual model developed for 8×8 DCT block to the 4×4 block used in H.264. In addition, we use P-frames to increase the watermark payload. The challenge in embedding the watermark in P-frames is that the video bit rate can increase significantly. By using the structure of the encoder, we significantly reduce the increase in video bit rate due to watermarking. Our method also exploits both temporal and texture masking.

We build a theoretical framework for watermark detection using a likelihood ratio test. This framework is used to develop two different video watermark detection algorithms; one detects the watermark only from watermarked coefficients and one detects the watermark from all the ac coefficients in the video. These algorithms can be used in different video watermark detection applications where the detector knows and does not know the precise location of watermarked coefficients. Both watermark detection schemes obtain video watermark detection with controllable detection performance. Furthermore, control of the detector's performance lies completely with the detector and does not place any burden on the watermark embedding system. Therefore, if the video has been attacked, the detector can maintain the same detection performance by using more frames to obtain its detection response. This is not the case with images, since there is a limited number of coefficients that can be watermarked in each image before the watermark is visible.

CHAPTER I

INTRODUCTION

The advent of digital television, the appearance of digital versatile disks (DVI), and the transfer of video files over the internet demonstrate the importance of digital video. Despite the success of MPEG-2 in the video coding industry, a new standard, H.264, with higher compression efficiency, is poised to replace it. As H.264 digital video becomes more prevalent, the industry will need copyright protection and authentication methods that are appropriate for it.

Since video signals are usually stored and distributed in a compressed format, it is often impractical to first decode the video sequence, embed a watermark, and then reencode it. An alternative approach is to embed the watermark in the compressed domain, which produces a lower-complexity video watermarking algorithm. Unfortunately, the large body of MPEG-2 video watermarking algorithms cannot be applied directly to H.264 because of the differences in the standards.

The goal of this thesis is to present a robust watermarking algorithm for H.264 and to address challenges in compressed-domain video watermarking. We give overviews of H.264 and digital watermarking in Section 1.1 and 1.2, respectively. Section 1.3 presents the organization of this thesis.

1.1 H.264 Standard

As with previous standards, H.264 does not explicitly define an encoder/decoder pair (codec), but defines the syntax of an encoded video bitstream and the method of decoding this bitstream. With the exception of the deblocking filter, most of the basic functional elements are present in the previous standards. The important differences in H.264 occur in the details of each functional block. We review some of these differences in the following subsections. More details can be found in [2, 45, 55, 73]. The block diagram of an H.264 encoder is shown in Figure 1.

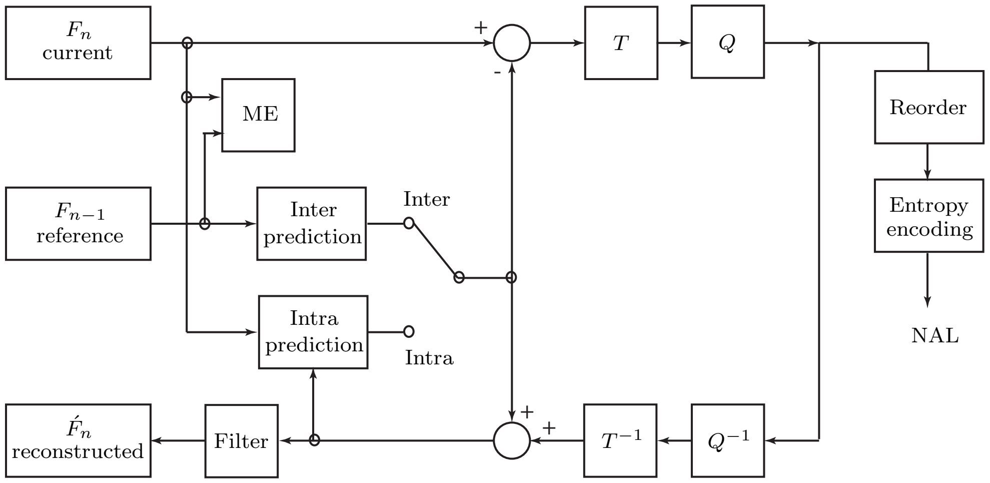


Figure 1: H.264 encoder block diagram.

1.1.1 Intra-Prediction

Unlike previous standards in which there is no prediction in their I-frames, I-frames of H.264 are predicted in an intra-prediction mode. Intra-prediction means that the samples of a macroblock are predicted by using only information of already transmitted macroblocks of the same frame. In H.264, two different types of intra-prediction, intra 4×4 and intra 16×16, are possible for the prediction of the luminance component Y. The intra 4×4 mode is based on predicting each 4×4 luma block separately and is well suited for coding the detailed areas of the frame. The intra 16×16 mode, on the other hand, performs prediction of the whole 16×16 luma block and is more suited for coding the smooth areas of the frame. There are nine different prediction modes in the intra 4×4 mode. One is dc prediction where one value is used to predict the whole 4×4 block. In addition to the dc prediction mode, there are eight other prediction modes, each for a specific prediction direction. All possible directions are shown in Figure 2. There are four different prediction modes for the intra 16×16: vertical prediction, horizontal prediction, dc prediction, and plane prediction. These prediction modes are shown in Figure 3.

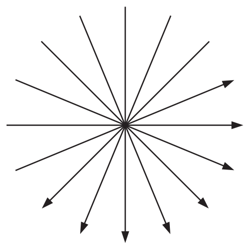


Figure 2: 4×4 luma prediction modes.

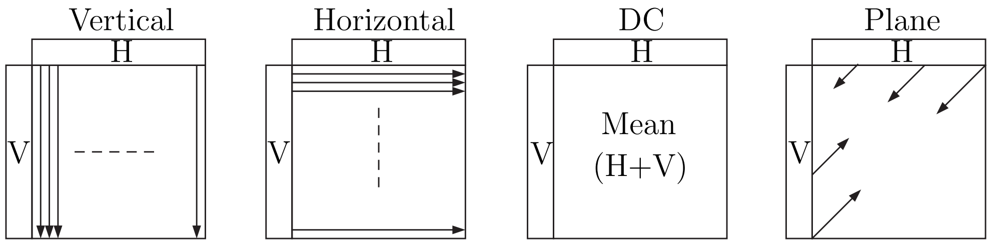


Figure 3: 16×16 luma prediction modes.

1.1.2 Inter-Prediction

Inter-prediction creates a prediction model from one or more previously encoded video frames or fields using block-based motion compensation. In previous standards, only 16×16 and 8×8 blocks are supported. However, in H.264, a range of block sizes (16×16, 16×8, 8×16, and 8×8) are supported. For an 8×8 sub-macroblock, one additional element specifies whether the corresponding 8×8 sub-macroblock is further divided into partitions with block sizes of 8×4, 4×8, or 4×4.

Each partition or sub-macroblock partition in an inter-coded macroblock is predicted from an area of the same size in a reference picture. The offset between the two areas (the motion vector) has quarter-sample resolution for the luma component and one-eighth-sample resolution for the chroma components. The luma and chroma samples at sub-sample positions do not exist and they are created using interpolation from nearby samples.

In previous standards, B-pictures are pictures that are encoded using both past and future pictures as references. The prediction is obtained by averaging the forward and backward prediction signal. However, H.264 uses a linear combination with arbitrary weights, regardless of the temporal direction. Furthermore, using H.264 it is possible to use images containing B-slices as reference images for further prediction, which was not possible in previous standards.

1.1.3 Transform Coding

Similar to previous standards, H.264 uses transform coding of the prediction residuals. Previous standards such as MPEG1 and MPEG2 apply an 8×8 two-dimensional discrete cosine transform (DCT). However, in H.264, an integer transform is applied to 4×4 blocks. H.264 uses three different types of transforms. The first one is applied to all 4×4 blocks of luminance and chrominance components of inter- or intra-macroblocks. The second transform is a Hadamard transform for the 4×4 array of luma dc coefficients of 16×16 intra-predicted macroblocks. The third transform is also a Hadamard transform for the 2×2 array of chroma dc coefficients. H.264 also has an option to use an 8×8 transform in addition to the 4×4 transform.

The H.264 4×4 integer transform is based on the DCT, but with several advantages:

- It is an integer transform; thus, all the operations can be carried out using integer arithmetic without loss of decoding accuracy. The core part of the transform can be implemented using only additions and shifts.
- The mismatch between the encoder and decoder is avoided. This has been a problem with the 8×8 DCT transform used in previous standards.
- The smaller 4×4 transform has visual benefits, resulting in less noise near edges.

1.1.4 Entropy Coding

H.264 specifies two alternative methods of entropy coding: a low-complexity technique based on context-adaptive variable length coding (CAVLC) and a more computationally demanding algorithm using context-based adaptive binary arithmetic coding (CABAC). Both methods represent major improvements in terms of coding efficiency compared to the statistical coding traditionally used in video coding standards.

1.1.5 In-Loop Deblocking Filter

The block-based structure of the H.264 architecture can cause severe blocking artifacts. H.264 defines an adaptive in-loop deblocking filter to reduce blocking distortion, where the strength of the filtering is adjustable. When the filtering process is carried out in the loop, the filtered image is used for motion-compensation prediction of future frames. This can improve compression performance because the filtered image is often a more faithful reproduction of the original frame than a blocky unfiltered image.

1.2 Digital Watermarking

Digital watermarking is a technique used for protecting the intellectual property rights of digital media owners. Watermarking embeds a signal into the data stream that is imperceptible to the human observer, but can be detected by a watermark detector, hence identifying the owner and possibly the customer to whom this copy was originally distributed. The watermark does not prevent a user from listening to, viewing, examining, or manipulating the content. One advantage of embedding the protection directly in the signal is that the protection cannot be removed without affecting the signal quality. While cryptographic techniques protect the data from eavesdroppers during transmission, digital watermarking was introduced to leave a mark in the signal to protect it after transmission. Thus, these two fields together provide complete protection of the digital data.

Watermarking is a special case of data hiding or steganography. Steganography is the practice of encoding secret information in a communication channel in a manner such that the existence of the information is concealed. Typically, in steganography, the secret information contains all the values, and the communication channel used to hide it is not of value itself. However, in digital watermarking, the secret message (watermark) is of little or no value on its own, and the communication channel (digital image and video) is of value.

1.2.1 Video Watermarking Applications

With the wide application of digital video, watermarking can add value to various video applications. Video watermarking applications are presented extensively in [16]. Some of these applications are summarized in Table 1.

Table 1: Applications of video watermarking.

Application	Purpose of the embedded watermark
Copy control	Prevent unauthorized copying.
Broadcast monitoring	Identify the video being broadcast and check usage.
Fingerprinting	Trace back a malicious user.
Video authentication	Insure that the original content has not been altered.
Copy protection	Prove ownership.
Enhanced video coding	Bring additional information e.g. for error correction.
Advertisement	Verify the frequency of display of an advertisement.
Content ID and archive	Add meta-data (e.g. owner, date, etc.) for archive.

1.2.2 Video Watermarking Requirements

When designing a watermarking algorithm, trade-offs exist among three parameters: payload, fidelity, and robustness. Data payload is the number of bits that can be embedded in the digital data, the fidelity is the degradation introduced into the signal, and the robustness is the ability of the watermark to remain readable after innocent or malicious signal processing operations on the signal. These parameters are conflicting, and they should be chosen to meet the requirements of the application.

When designing watermarking algorithms for video, there are additional conflicting parameters, such as the need for low complexity and constant bit rate. In some video applications, watermark embedding or detection needs to be performed in real time. Thus, video watermarking algorithms should have low complexity. Because of the large size of video files, they are usually stored and distributed in a compressed format. Video watermarking algorithms should not increase the bit rate of the compressed video.

1.2.3 Video Watermarking Attacks

There are many challenges when designing a video watermarking algorithm. Simple signal processing enhancement techniques, such as gamma correction, sharpening, and filtering, and geometric attacks, such as cropping, resampling, and rotation, alter the performance of watermarking algorithms. Transcoding, which involve changing the compression ratio to adapt to the storage capacity, converting the video format, and chrominance resampling, are likely to remove the watermark. Spatial desynchronization, such as changes in display formats (4/3, 16/9, and 2.11/1) and changes of resolution (NTSC, PAL, and SECAM), and temporal desynchronization such as frame rate modification may also affect watermark detection algorithms. Also, video editing, such as the addition of a commercial into the middle of a movie, a transition between scenes, and superimposition, such as picture-in-picture technology, subtitles and logos, degrade the performance of watermarking algorithms.

A more serious problem with video or audio signals, which are long, is the possibility of a self-collusion attack. A collusion attack is a very powerful attack for still images. There are two types of collusion attacks. If the same watermark is embedded in different data, the watermark data can be estimated from each occurrence and the average of those estimates will be a refined estimate. If different watermarks are embedded in the same data, several users can collude by averaging their decoded signals to reduce the strength of the watermark and possibly render it unreadable. However, with video one video sequence is enough to remove the watermark. If the same watermark is embedded in all the frames, the first type of collusion can be used to remove the watermark from different scenes. If a different watermark is embedded in each frame, the second type of collusion can be used to remove the watermark from correlated scenes. Recognizing these possibilities, the watermarks inserted in two video frames should be as similar as the two frames are.

1.2.4 Classifications of Watermarking Techniques

In terms of their robustness to attacks, watermarking techniques can be classified as fragile, robust and semifragile. Fragile watermarks do not survive lossy transformations to the original host signal; their purpose is tamper detection of the original signal. Placing the watermark information into the perceptually insignificant portions of the data guarantees imperceptibility, but provides a fragile watermark. Robust watermarks are used for security applications and copyright protection. The technical challenge is to provide transparency and robustness, which are conflicting requirements. For a watermark to be robust, the watermarks should be embedded into the significant portions of the data. Semifragile watermarks should be insensitive to some common innocent transformations, such as compression, but should be sensitive to image transformations that alter the information, such as replacing a portion of the image. The challenge for semifragile watermarking from a signal processing perspective is to provide a watermark that can distinguish between information altering and simple signal processing transformations.

Watermarking techniques are also classified as public and private. Public or blind watermarking algorithms do not require the original image to detect the watermark.

Public watermarks are typically used for applications requiring a robust watermark, such as identifying the buyers to prevent illegal duplication and distribution. Private or non-blind watermarking algorithms do require the original image to verify the watermark. Private watermarks are necessary for some fragile watermarking applications, such as authentication and tamper detection.

In terms of the domain in which the watermark is inserted, watermarking techniques can be classified as spatial-domain, transform-domain or compressed-domain watermarking algorithms. We will expand on these algorithms below. An overview of a large number of watermarking techniques in the different domains can be found in [18].

1.2.4.1 Watermarking in the Spatial Domain

In spatial domain watermarking systems, the watermark is embedded directly in the spatial domain (pixel domain). Many of the spatial watermarking techniques provide simple and effective schemes for embedding an invisible watermark into an image, but are less robust to common attacks. Watermarking schemes in the spatial domain are in general less robust toward noise-like attacks, such as lossy JPEG compression. However, a big advantage is that the watermark may easily be recovered if the image has been cropped or translated. Here we summarize a small portion of the proposed spatial domain methods. More details about these algorithms can be found in a review paper by Hartung *et al.* [18]. The abundance of spatial-domain methods results from their simplicity and efficiency.

Tanaka *et al.* introduced the idea of tagging images to secretly hide information and assure ownership rights first in 1990 [64, 65]. Then, in 1993 Caronni [11] described an overall system to track unauthorized image distribution. He proposed marking images using spatial signal modulation and called the process tagging. A tag is a square with a constant value proportional to the maximum image brightness within the square and decaying outside the border. A selected image area is tagged by adding or subtracting the tag and a random, zero mean, noise pattern. In the same year, Tirkel *et al.* recognized the importance of digital watermarking and possible applications for image tagging, copyright enforcement, counterfeit protection, and controlled access to image data [67]. In their approach, the watermark in the form of an m-sequence-derived PN code is embedded in the least significant bit (LSB) plane of the image data. This method is actually an extension to simple LSB coding schemes in which the LSBs are replaced by the coding information. The idea of using m-sequences and LSB addition was extended and improved by the authors through the use of two-dimensional m-sequences, which resulted in more robust watermarks [68]. A modified version of the method was presented by Schyndel *et al.* in [70] explicitly mentioning the term digital watermarking. About the same time Matsui and Tanaka proposed several watermarking techniques [34]. Their first method is based on a predictive coding scheme for gray scale images; their second method modifies the ordered dithering scheme for binary pictures; and their third scheme embeds the watermark in facsimile documents.

Since the above techniques were introduced, interest and research activities in watermarking have increased significantly. In some recent work, Bender *et al.* proposed two methods for data hiding [5]. In the first method, called "Patchwork," randomly selected pairs of pixels are used to hide 1 bit by increasing one pixel by one and decreasing the other pixel by one. In the second approach, called "Texture Block Coding," the watermark is embedded by copying one image texture block to another area in the image with a similar texture. To recover the watermark, the autocorrelation function has to be computed. Pitas *et al.* proposed signature casting on digital images [36, 48, 49], which is based on the same basic idea as the patchwork algorithm proposed by Bender *et al.* in [5]. Langelaar *et al.* proposed an improved version of this idea in [29, 30]. The image is tiled into square blocks with a size being a multiple of eight. A single bit is embedded by iteratively modifying a pseudorandomly selected block. To increase the performance of spread-spectrum watermarking in the spatial domain, Kutter *et al.* proposed a method which exclusively works with the blue image component (in the RGB color space) to maximize the watermark strength, while keeping visual artifacts minimal [27]. Extensions to this method allow increased robustness and even watermark recovery after geometrical attacks and printing-scanning [26]. Macq *et al.* introduced watermarking adapted to the human visual system (HVS) using masking and modulation [14, 15]. In their scheme, the watermark in the form of a spatially limited binary pattern is low-pass filtered, frequency modulated, masked, and then added to the host image. Wolfgang *et al.* proposed a watermarking technique to verify image authenticity [74, 75] based on an approach similar to the m-sequence approach suggested by Schyndel *et al.* for the one-dimensional case [70] and Tirkel *et al.* for the two-dimensional case [68].

Watermark embedding based on quantization has been proposed by Chen and Wornell [12]. Their method is called quantized index modulation (QIM) and is based on a set of N-dimensional quantizers. Maes *et al.* proposed modifying geometric features of the image [32]. The method is based on a dense line pattern, generated pseudorandomly and representing the watermark. Fractal image compression, an idea similar to spatial domain watermarking, was first proposed in [53]. In fractal image compression, the image is coded using the principles of iterated function systems and self similarity [56]. A drawback of this technique is the slow speed caused by the fractal compression scheme.

1.2.4.2 Watermarking in the Transform Domain

In transform domain watermarking systems, watermark insertion is done by transforming the image into the frequency domain using a discrete Fourier transform (DFT), full-image DCT, block-wise DCT, wavelet, Hadamard, Fourier-Mellin, or other transforms. It is often claimed that embedding in the transform domain is advantageous in terms of visibility and security. It has been shown that for maximum robustness, watermarks should be embedded into the same spectral components that the host data already populate. For images and videos, these are typically the low frequencies. Designing watermarking algorithms in the transform domain is not as simple as in the spatial domain. However, there are many block DCT-domain algorithms, because this transform is used by many compression standards such as JPEG, MPEG2, and H.263, and etc.

Efficient watermarking in the DCT domain was first introduced by Koch *et al.* [10, 23, 24]. As in the JPEG compression scheme, the image is first divided into square blocks of size 8×8 for which the DCT is computed. From a pseudorandomly selected block, a pair of midfrequency coefficients is selected from 12 predetermined pairs. To embed a bit, the coefficients are then modified such that the difference between them is either positive or negative, depending on the bit value. Bors and Pitas [8, 9] suggested a method that modifies DCT coefficients satisfying a block site selection constraint. The image is first divided into blocks of size 8×8. Certain blocks are then selected according to a Gaussian network classifier decision. The middle range frequency DCT coefficients are then modified, using either a linear DCT constraint or a circular DCT detection region, to convey the watermark information. Swanson *et al.* [61, 62] suggested a DCT-domain watermarking technique, based on frequency masking of DCT blocks, which is similar to methods proposed by Smith and Comiskey [58]. Tao and Dickinson [66] introduced an adaptive DCT-domain watermarking technique based on a regional perceptual classifier with assigned sensitivity indexes. Podilchuk *et al.* [52, 51] introduced perceptual watermarking using the just noticeable difference (JND) to determine an image-dependent watermark modulation mask. The watermark is embedded into selected coefficients in either the DCT or wavelet transform domain. For DCT coefficients, the author suggests using a perceptual model defined by Watson, based on utilizing frequency and brightness sensitivity as well as local contrast masking. This model provides image-dependent masking thresholds for each 8×8 DCT block. Piva *et al.* described another DCT-based method which exploits the masking characteristics of the HVS [50].

Frequency-domain watermarking was first introduced by Boland *et al.* [7] and Cox *et al.* [13], who independently developed perceptually adaptive methods based on modulation. Cox *et al.* drew parallels between their technique and spread-spectrum communication since the watermark is spread over a set of visually important frequency components. The watermark consists of a sequence of numbers $x = x_1, ..., x_n$ with a given statistical distribution, such as a normal distribution $N(0, 1)$ with zero mean and variance one. The watermark is inserted into the image V to produce the watermarked image V'. Three techniques are proposed for watermark insertion, but the one most commonly used is

$$v_i' = v_i(1 + \alpha x_i) \tag{1}$$

where α determines the watermark strength and the v_is are perceptually significant spectral components. The scheme can be generalized by introducing multiple scaling parameters α_i to adapt to the different spectral components and thus reduce visual artifacts. To verify the presence of the watermark, the similarity between the recovered watermark, given by the difference between the original image and the possibly tampered image, and the original watermark, is measured.

Ruanaidh *et al.* proposed watermarking by modification of the phase in the frequency domain [43]. To embed a bit the phase of a selected coefficient of an $N_1 \times N_2$, DFT is modified by adding a small δ. The phase must satisfy negative symmetry for the watermarked image to be real, which leads to the additional modification. In another publication, Ruanaidh *et al.* explicitly design a watermarking technique invariant to translation, rotation, and scaling [44]. The method is a hybrid between DFT and log-polar mapping. A variation of their idea based on the Radon transform was proposed by Wu *et al.* [78].

Embedding the watermark using a multiresolution decomposition has first been proposed by Boland *et al.* [7]. As for schemes working in other transform domains, the watermark is usually given by a pseudo-random 2-D pattern. Both the image and watermark are decomposed using a 2-D wavelet transform, and in each subband of the image a weighted version of the watermark is added. Watermark decoding is, as usual, based on a normalized correlation between the estimate of the embedded watermark and the watermark itself. Various wavelet-based schemes have been proposed [21, 25, 79, 81]. The differences between the schemes usually lie in the way the watermark is weighted in order to decrease visual artifacts.

1.2.4.3 Compressed-Domain Video Watermarking

Since video signals are usually stored and distributed in a compressed format, it is often impractical to first decode the video sequence, embed the watermark, and then reencode it. Thus, designing low-complexity video watermarking algorithms that embed the watermark in the compressed domain is attractive. Most of the previous work on compressed-domain video watermarking focused on embedding the watermark into the MPEG2 bitstream. The residual blocks in the MPEG2 standard are compressed using the DCT transform, quantized, reordered, run-level coded, and then variable length coding is applied.

Langelaar *et al.* proposed two real-time watermark embedding methods [28]. Both methods embedded the watermark directly into the MPEG compressed bitstream. The first method embedded the watermark by changing the variable length codes (VLCs). The watermark is embedded by selecting suitable VLCs and forcing their least-significant bits (LSB) to match the corresponding watermark bits. The second method discarded some of the high-frequency DCT coefficients of the bitstream to embed the watermark. Hartung *et al.* proposed a method for embedding the watermark in the uncompressed video and an extension of that method for embedding the watermark in the MPEG2 video [17]. The authors' method, which is based on spread spectrum watermarking [13], spread the watermark bits by a large factor called the chip rate to obtain the spread sequence. The spread sequence is amplified with an adjustable factor and is modulated by a binary pseudo-noise sequence and added to the video signal. Alattar *et al.* proposed an MPEG4 compressed-domain video watermarking method, and its performance is studied at low video bit rates [3]. This approach is similar to the approach in [17]; however, the authors used a synchronization template to combat geometric attacks. Their method also featured a gain control algorithm that adjusts the embedding strength of the watermark, depending on local image characteristics. Simitopoulos *et al.* proposed another compressed-domain video watermarking algorithm that operates directly on the MPEG2 bitstream [57]. The proposed algorithm altered only the quantized ac coefficients of luminance blocks that belong to intra-frames. Perceptual models combining perceptual analysis and block classification are used during the embedding to preserve the video quality. In [76], Wolfgang *et al.* proposed an image adaptive DCT-based (IA-DCT) approach that used the visual model described in [72]. This model consists of an image-independent part based on frequency sensitivity and an image-dependent part based on luminance sensitivity and contrast masking. They have extended their IA-DCT technique to video. The best visual quality was obtained by using the IA-DCT watermarking technique at every I-frame and applying a simple linear interpolation of the watermarks to the frames between two consecutive I-frames.

A few recently published papers have concentrated on embedding a watermark in the H.264 bitstream sequence. Qiu *et al.* proposed a hybrid watermarking scheme that embedded a robust watermark in the DCT domain and a fragile watermark in the motion vectors [54]. Their technique embeds the watermark in the compressed

H.264 video, but it is not robust against common watermarking attacks. Wu *et al.* presented a blind watermarking algorithm by embedding the watermark in the H.264 I-frames [77]. Their scheme survives H.264 compression attacks with more than a 40:1 compression ratio in I-frames. However, their scheme requires decompressing the video to embed the watermark.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 presents a watermarking algorithm that is robust to self-collusion attacks. Chapter 3 introduces a robust watermark embedding method for H.264 using Watson's human visual model. In Chapter 4, we build a theoretical framework for watermark detection based on a likelihood ratio test. This framework is used to obtain video watermark detection with controllable detection performance when the precise location of the watermark is known to the detector, a detector we call location-aware. In Chapter 5, we introduce a watermark embedding algorithm that controls the video bit rate increase in compressed-video. This algorithm makes watermark embedding in P-frames possible. Chapter 6 presents a variation on the watermark detection algorithm presented in Chapter 4 that does not depend on where the watermark signal is embedded called the location-unaware detector. Finally, Chapter 7 summarizes this work and suggests future research directions.

CHAPTER II

SELF-COLLUSION RESISTANT WATERMARK EMBEDDING

Watermarking digital video introduces challenges that are not present when watermarking digital images. The large amount of data and inherent redundancy between frames makes video watermarking algorithms susceptible to self-collusion attacks. The self-collusion attack is one of the most powerful attacks for video. If the same watermark is embedded in all the frames, the watermark data can be estimated from each frame, and the average of those estimates will be a refined estimate. If different watermarks are embedded in similar frames, the attacker can average those frames to reduce the strength of the watermark and possibly render it unreadable. Therefore, the watermarks inserted in two video frames should be as similar as the two frames are.

The collusion problem was first addressed by Swanson *et al.* [63], who presented a scene-based video watermarking technique that is robust to self-collusion attacks. In their technique, the video sequence is segmented to different scenes and a temporal wavelet transform is applied to the frames in each scene. The watermark is added to the low-pass and high-pass frames of the temporal wavelet transform. They compute the 8×8 DCT of those frames and use the perceptual masking properties of the human visual model to embed an invisible and robust watermark. Their method uses a two-level hierarchy of transforms and is considered to be highly complex. Recent work by Trappe *et al.* [69] has focused on collusion-resistant digital fingerprinting that can identify colluders; their work makes use of effective anti-collusion codes for CDMA-type watermarking using the theory of combinatorial designs. In [60], Su *et* *al.* present a theoretical framework for the linear collusion analysis of watermarked digital video sequences, and derive a new theorem equating a definition of statistical invisibility, collusion-resistance, and two practical watermark design rules. In [59], the authors develop a novel video watermarking framework based on their collusion-resistant design rules formulated in [60]. They propose employing a spatially-localized image dependent approach to create a watermark whose pairwise frame correlations approximate those of the host video. To characterize the spread of its spatially-localized energy distribution, the notion of a watermark footprint is introduced. They explain how a particular type of image dependent footprint structure, comprised of subframes centered around a set of visually significant anchor points, can lead to two advantageous results: pairwise watermark frame correlations that more closely match those of the host video for statistical invisibility, and the ability to apply image watermarks directly to a frame sequence without sacrificing collusion-resistance.

In this chapter, we design a novel low complexity watermarking algorithm that is robust to self-collusion attacks [38]. The algorithm embeds the watermark into the quantized ac residuals of the H.264-compressed video. It achieves collusion resistance by embedding the watermark in the same location in similar frames and different locations in dissimilar frames. The coefficient within a macroblock that holds the watermark is determined by a key that is specific to that macroblock. This requires a long key stream sequence. To avoid this problem, the key is generated using a public key extracted from some features of the macroblock and the copyright owner's secret key. It is proposed that the relative difference of the dc coefficients of the 4×4 blocks in a macroblock is a robust feature for public key extraction.

2.1 Proposed Method

Our proposed H.264 watermarking algorithm works at the macroblock level of I-frames. A macroblock contains a 16×16 sample region of a video frame. I-frames are chosen for watermark embedding because their existence is crucial for the video signal. Also, P- and B-frames are highly compressed by motion compensation and there is less capacity for embedding a watermark in them. The luminance component of macroblocks in an I-frame is intra-coded in 16×16 or 4×4 intra-prediction modes. Each 4×4 block of residual data is transformed by an integer transform after intra-prediction. If the macroblock is coded in the 16×16 intra-prediction mode, the dc coefficients of all 4×4 blocks are transformed by a 4×4 Hadamard transform after the 4×4 integer transform to decorrelate these coefficients further. We only embed the watermark in the quantized ac residuals of the luma component of 4×4 intra-predicted macroblocks. We do not embed the watermark in the 16×16 intra-predicted macroblocks for two reasons. First, the 16×16 intra-prediction mode is used for smooth regions of the frame, and watermark embedding causes visible artifacts there. Second, the extra Hadamard transform for this macroblock decorrelates the dc coefficients even more, and many of these coefficients are zero.

Our proposed algorithm embeds the watermark in one quantized ac residual of a macroblock. The security of the algorithm is based on the randomness of the selected coefficient for watermark embedding. Embedding the watermark in only one coefficient in a macroblock does not induce visible artifacts. However, the attacker cannot identify which coefficient has been selected. Therefore, he/she has to change at least half of the coefficients to make watermark detection impossible. However, changing half of the coefficients results in visible artifacts in the video signal and renders the video useless.

The selection of the coefficient in the i^{th} macroblock for watermark embedding is under the control of a key. If the same key is used for every frame, the watermarking algorithm becomes vulnerable to a self-collusion attack. Thus, a very long key stream sequence is required. Transmitting a long key, however, would make the algorithm impractical. This problem is solved by generating the key from a combination of a public key, Kp_i, extracted from some features of the macroblock, and a secret key, Ks, possessed by the copyright owner. The public key is extracted from each macroblock and passed as the plaintext to a cryptographic system with the secret key, Ks. The ciphertext generated by the cryptographic system is the key for that macroblock. Since the security demands of watermarking systems are less than those of cryptographic systems, a fast and simple cryptographic scheme can be used for this purpose. We used a shift cipher with modulus 2, key Ks, and plaintext Kp_i. Two bits of this key determine the selected 8×8 block in the macroblock, $b8_i$, another two bits determine the selected 4×4 block in the 8×8 block, $b4_i$, and four bits determine the selected ac quantized residual in that 4×4 block, cw_i, for watermark embedding. Kp should be extracted from some features of the macroblock that cannot be changed by the attacker without degrading the perceptual quality of the video. In the next section, we describe the public key extraction procedure. Figure 4 shows the structure of our proposed watermarking algorithm.

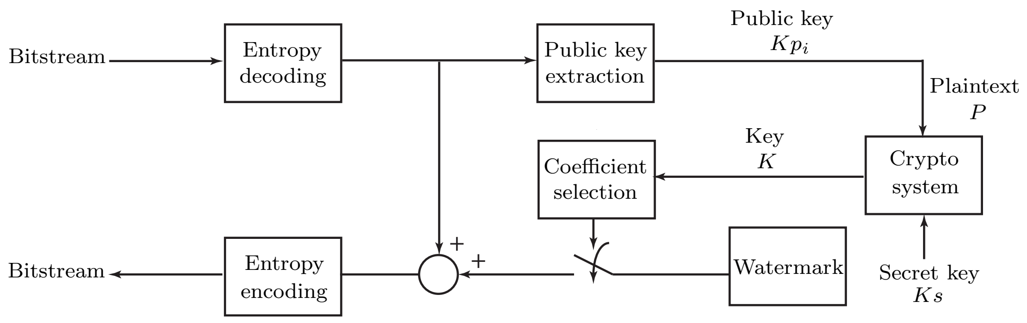


Figure 4: A watermark embedding algorithm robust to self-collusion attack.

2.1.1 Extracting the Public Key

To make the extracted public key robust, a feature of the macroblock should be used to which the human eye is sensitive. One such feature is the set of dc coefficients of the 4×4 blocks. If the dc coefficients themselves are used for public key extraction, the attacker could change the dc coefficient of every block by the same amount, which would make watermark detection impossible. This would result in a darker or brighter frame, but the perceptual quality of each frame would be preserved. However, if the relative difference of the dc coefficients of 4×4 blocks is used to determine the public key, the attacker has to increase or decrease the dc coefficient of one block or more to make the public key extraction impossible for the copyright owner. This results in visible artifacts.

To determine the public key, Kp_i, the quantized dc residuals of 4×4 blocks of the i^{th} 4×4 intra-coded macroblock are extracted and put in a vector of dc coefficients, DC_i, in a key-scrambled fashion. DC_i can have up to 24 elements, 16 elements from the luma component, Y, and four components from each of the chroma components, Cb and Cr. The structure of the public key generation is shown in Figure 5. The j^{th} bit of Kp_i is derived from DC_i as follows:

$$Kp_i(j) = \begin{cases} 0 & DC_i(j) > DC_i(j+1) \\ 1 & DC_i(j) \leq DC_i(j+1) \end{cases} \tag{2}$$

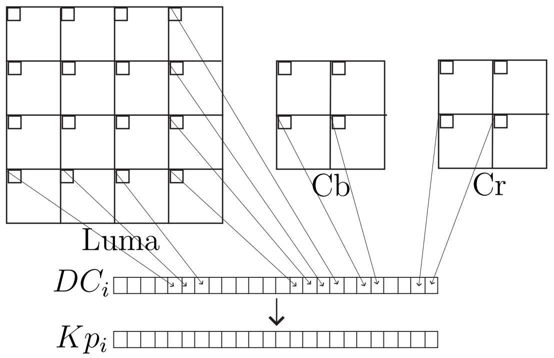
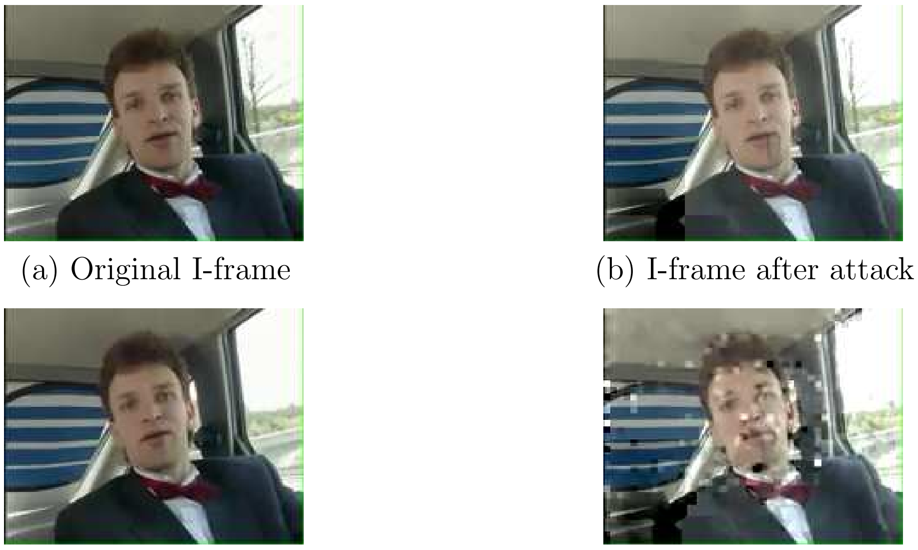


Figure 5: Structure of public key generation.

To show the robustness of the extracted public key, one of the dc coefficients in a macroblock is changed to erase one bit of Kp_i. The resulting I-frame and one of the following P-frames are shown in Figure 6. It can be observed that changing only one dc coefficient in a macroblock lowers the perceptual quality of the video.


(a) Original I-frame (b) I-frame after attack

(a) Original P-frame (b) P-frame after attack

Figure 6: Robustness of the extracted public key.

2.1.2 Watermark Embedding

If a compressed video bitstream is to be watermarked, it has to be decoded to some extent. The closer the watermark embedding operation is to the entropy coding level, the less computationally complex and more suitable for real-time application the algorithm becomes. However, the closer the embedding is to the DCT transform operation, the less the degradation induced by watermark embedding becomes. We embed the watermark in the reordered quantized ac residuals. Because quantization is a lossy operation, it is desirable to embed the watermark after quantization to avoid possible erasure of the watermark. Furthermore, entropy coding and decoding are fast procedures, and watermark embedding and detection can be done in real time.

As discussed in Subsection 2.1.1, several bits of Kp_i are used to select the coefficient, cw_i, in the macroblock for watermark embedding. Kp_i has more bits than required. Thus, the extra key bits can be used to make the algorithm more secure and robust. To embed the watermark W_i in macroblock i, cw_i is modified as follows:

if $W_i = 0$,

$$cw_i = \begin{cases} cw_i - 1 & \text{if } cw_i \mod 2 = 1 \\ cw_i & \text{if } cw_i \mod 2 = 0 \end{cases} \tag{3}$$

if $W_i = 1$,

$$cw_i = \begin{cases} cw_i & \text{if } cw_i \mod 2 = 1 \\ cw_i - 1 & \text{if } cw_i \mod 2 = 0 \end{cases} \tag{4}$$

The maximum change made to the quantized coefficient selected for watermark embedding is one level. Thus, the modification of the unquantized DCT coefficient is as large as the quantization step size for that block. Consequently, the degradation induced by watermark embedding is proportional to the quantization error.

Entropy coding produces short codewords for frequently occurring values and longer codewords for less frequently occurring values. Generally, the values closer to zero occur more frequently. The watermark embedding method moves some values closer to zero and some values further. Thus, the average bit rate remains more or less the same. There is a coded block pattern parameter in H.264 that indicates which blocks within a macroblock contain coded coefficients. This parameter is computed before watermark embedding. Thus, if the watermarking algorithm selects a block of all zero coefficients to embed the watermark, this parameter does not let the change take place and helps to control the bit rate. When the detector finds that the coefficient selected to hold the watermark bit is in an all-zero block, it knows that the watermark has not been embedded in that macroblock.

2.1.3 Watermark Decoding

Watermark decoding is performed after entropy decoding. The decoder uses the dc coefficients of the macroblock with its secret key to find the location of the watermark. The watermark bit in the macroblock is determined as follows:

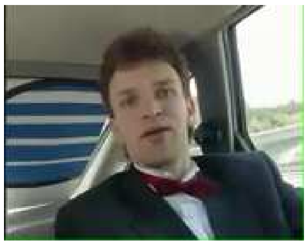
$$\hat{W}_i = \begin{cases} 0 & \text{if } cw_i \mod 2 = 0 \\ 1 & \text{if } cw_i \mod 2 = 1 \end{cases} \tag{5}$$

2.2 Simulation Results

Our proposed watermarking algorithm was implemented in the H.264 reference software version JM86 [1]. The standard video sequence CARPHONE (QCIF, 176×144) at the rate of 30 frames per second was used for simulation. Figure 7 shows the fidelity of our proposed watermarking algorithm. The frames on the left are an I-frame and one of the following P-frames of this video sequence coded with the H.264 codec. The frames on the right are the watermarked version of these frames.

(a) Original I-frame

(b) Watermarked I-frame


(a) Original P-frame

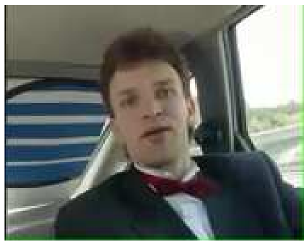

(b) Watermarked P-frame

Figure 7: Fidelity of the self-collusion resistant watermark embedding algorithm.

The number of watermark bits embedded in an I-frame of six standard video sequences, the percentage of watermark bits recovered after an H.264 reencoding attack, and the percentage increase in the video bit rate after watermarking are given in Table 2. The reencoding attack encodes the watermarked video again with an H.264 encoder, which has the same encoding parameters as the original encoder

Table 2: Number of watermark bits per frame, watermark recovery rate after reencoding attack, and percentage increase in video bit rate of collusion resistant watermarking algorithm for six standard video sequences.

Video sequence	Watermark bits per frame	Reencoding recovery rate	Bit rate increase
CARPHONE	54	68.42%	0.82%
CLAIRE	28	88.23%	1.53%
MOBILE	90	84.70%	0.22%
MOTHER	48	76.47%	1.05%
TABLE	35	82.75%	0.21%
TEMPETE	80	86.30%	0.32%

used to watermark the video, and decodes the watermark in the H.264 decoder. On average, the watermarking process increased the size of the compressed video by only 0.69%.

Two examples of a hypothetical adversary's attempt to remove the watermark are shown in Figure 8. One ac residual and half of the ac residuals in each 4×4 block were modified in the frame on the left and right, respectively. While the first attack only erases a few bits of the watermark, the second one erases half of the watermark bits. Both attacks result in a significant degradation in video quality. The proposed algorithm is not robust against signal processing attacks, because any simple signal processing operation, such as filtering, changes the prediction modes and subsequently the residuals in the I-macroblocks of H.264. However, the algorithm is robust against modifications in the H.264 bitstream domain.

2.3 Conclusion

In this chapter, a novel, low complexity watermarking algorithm robust to self-collusion attack was presented. The coefficient in a macroblock to be embedded with the watermark is selected by a key. The key is generated from a robust feature extracted from the macroblock and a secret key possessed by the copyright owner.

(a) One coefficient (b) Half of the coefficients

Figure 8: Examples of adversary's attempt to remove the watermark by modifying a certain number of quantized ac residuals in a 4×4 block.

The relative difference of the dc coefficients of 4×4 blocks is used as a robust feature. Using macroblock features for watermark embedding makes the algorithm more robust against self-collusion attacks by embedding the watermark in the same location in similar frames and different locations in dissimilar frames. The algorithm is fast and appropriate for real-time applications. Simulation results show that watermark embedding preserves the perceptual quality of the video. On average, the proposed algorithm increased the video bit rate 0.69%. The algorithm is robust against modification in H.264 bitstream domain, however, it is not robust against signal processing attacks, because they change the prediction modes and subsequently the residuals in the I-macroblocks of H.264. The result of this work appears in [38].

CHAPTER III

PERCEPTUAL WATERMARK EMBEDDING

The main challenge in designing digital watermarking schemes is to balance transparency, robustness and payload, which are conflicting parameters. When embedding a watermark in compressed videos, it is desirable to obtain transparency while embedding the largest number of watermark bits with the maximum watermark strength. Human visual models are needed to determine a perceptual upper bound on the watermark signal strength in each portion of the signal.

Different techniques have been developed that incorporate perceptual knowledge into watermarking schemes to provide robustness and transparency. Wolfgang *et al.* used the visual models developed in [72] for still images to find the just noticeable difference (JND) for each coefficient, based on frequency sensitivity, luminance masking, and contrast masking [76]. They embedded the watermark only in those coefficients that exceeded their JNDs. They extended their method to video by using the algorithm on every I-frame and applying linear interpolation of the watermarks to the frames between consecutive I-frames. Simitopoulos *et al.* proposed a watermarking scheme that embeds the watermark directly into the MPEG stream [57]. To make the watermark imperceptible, perceptual analysis [72] and block classification techniques were combined.

In this chapter, we develop a perceptual watermarking algorithm for H.264 that is robust to common signal processing attacks [39, 41]. Since H.264's high compression performance leaves little room for an imperceptible signal to be inserted, we employ a human visual model to increase the payload and add robustness while limiting visual distortion. Watson *et al.* derived a model for distortion perception in 8×8

DCT blocks [46, 72]. This perceptual model has been used in [57, 76] to design watermarking algorithms for still images and MPEG-2 video. However, H.264 uses a 4×4 transform [33] instead of an 8×8 transform. The 4×4 transform is an integer orthogonal approximation to the DCT. Since the transform is defined by exact integer operations, inverse-transform mismatches are avoided. In this work, we extend the human visual model to the 4×4 DCT block. If all the coefficients with visual capacity for watermark embedding are used, the visual quality of the video will be degraded. We propose embedding the watermark in a selected subset of the coefficients that have visual watermarking capacity by using a key-dependent algorithm. This makes the algorithm more robust to malicious attacks. Furthermore, we design our algorithm so that the watermark is spread over frequencies and blocks. This reduces the error pooling effect described by Watson [72]. Error pooling has not been considered in previous perceptual watermarking algorithms [57, 76].

The algorithm we proposed in Chapter 2 embeds the watermark in the compressed video, but this algorithm is not robust against common watermarking attacks. This is because the watermark is embedded in and extracted from the I-frame residuals. Any simple processing, such as filtering followed by reencoding by an H.264 encoder changes the intra-macroblock prediction modes, and thus the residuals, which makes watermark recovery impossible. In this chapter, we present a robust watermarking algorithm for H.264. To achieve this goal we embed the watermark in the quantized DCT residuals to avoid decompressing the video and also to reduce the complexity of the watermarking algorithm. However, the watermark is extracted from the decoded video sequence to make the algorithm robust to intra-prediction mode changes.

3.1 A Human Visual Model for the 4×4 *DCT*

The DCT coefficients $X_{i,j}$ of an $M \times M$ block of image pixels $x(n_1, n_2)$ expand the block in terms of the DCT basis functions as follows:

$$x(n_1,n_2) = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} X_{i,j} c_i c_j \cos\left(\frac{\pi(2n_1+1)i}{2M}\right)\cos\left(\frac{\pi(2n_2+1)j}{2M}\right), \tag{6}$$

where

$$c_i = \begin{cases} \sqrt{1/M} & i = 0 \\ \sqrt{2/M} & i > 0. \end{cases} \tag{7}$$

Human visual sensitivity for each DCT basis function varies as a function of its frequency. Peterson *et al.* measured quantization error thresholds at various DCT frequencies in an 8×8 DCT block [47]. Here, we extend the quantization error visibility thresholds for an 8×8 DCT block to those appropriate for a 4×4 DCT block.

The basis functions of a 4×4 DCT are defined as

$$c_{i_4} c_{j_4} \cos\left(\frac{\pi(2n_1+1)i_4}{2 \times 4}\right)\cos\left(\frac{\pi(2n_2+1)j_4}{2 \times 4}\right) \qquad 0 \le i_4, j_4 \le 3, \tag{8}$$

and the basis functions of an 8×8 DCT are defined as

$$c_{i_8} c_{j_8} \cos\left(\frac{\pi(2n_1+1)i_8}{2 \times 8}\right)\cos\left(\frac{\pi(2n_2+1)j_8}{2 \times 8}\right) \qquad 0 \le i_8, j_8 \le 7. \tag{9}$$

Comparing equations (8) and (9) suggests that the basis function $i_4 j_4$ of a 4×4 DCT will have the same frequencies as the basis function $i_8 j_8$ of an 8×8 DCT, if

$$\begin{aligned} i_8 &= 2 \times i_4 \\ j_8 &= 2 \times j_4 \end{aligned} \tag{10}$$

hold. Since for all $0 \le i_4, j_4 \le 3$, there exists an i_8 and j_8 in the range $0 \le i_8, j_8 \le 7$, the visibility thresholds of a 4×4 DCT basis function can be derived from the known visibility thresholds of an 8×8 DCT basis functions. The factors c_i and c_j cause the amplitude of the errors for a 4×4 DCT to be twice the size as those of an 8×8 DCT. Thus, to obtain invisibility, the visibility threshold of the $i_4 j_4$ basis function of a 4×4 DCT is obtained by dividing the visibility threshold of the $2i_4 2j_4$ basis function of an 8×8 DCT by 2. We used the quantization matrices given in [47] for an 8×8 DCT block in the YC_bC_r color space to derive the quantization matrices for a 4×4 DCT block. The actual visibility threshold $t_{i,j}$ is half the quantization step size $q_{i,j}$.

To obtain an image-dependent quantization matrix, two other effects from [72], luminance masking and contrast masking, are exploited. A simple solution to approximate luminance masking is with a power function as

$$t_{i,j,k} = t_{i,j}(X_{0,0,k}/\bar{X}_{0,0})^{a_T}. \tag{11}$$

$\bar{X}_{0,0}$ is the dc coefficient corresponding to the mean luminance of the display and $X_{0,0,k}$ is the dc coefficient of block, k. a_T controls the degree to which this masking occurs. We choose $a_T = 0.649$ as suggested in [72]. Finally, contrast masking gives the masked threshold, $m_{i,j,k}$, for a DCT coefficient $X_{i,j}$ of block k ($X_{i,j,k}$) as

$$m_{i,j,k} = \max[t_{i,j,k}, |X_{i,j,k}|^{w_{i,j}} {t_{i,j,k}}^{1-w_{i,j}}] \tag{12}$$

where $w_{i,j}$ is between 0 and 1. We choose $w_{i,j} = 0.7$ as the authors of [72] recommend.

The image-dependent approach ensures that each error falls below a threshold. Furthermore, Watson noted that the visibility of an error is not based solely on the visibility of the largest error, but instead reflects a pooling of errors over frequency and within a block. We spread the watermark over frequencies and blocks to reduce error pooling, which has not been done in the previous perceptual watermarking algorithms [57, 76].

3.2 Proposed Method

We compute the masked error visibility threshold, $m_{i,j,k}$, for each coefficient $X_{i,j}$ in block k. This threshold is divided by the H.264 quantization step size for that block,

Q_k, to determine the capacity of that coefficient, $s_{i,j,k}$, for holding the watermark:

$$s_{i,j,k} = \text{floor}(m_{i,j,k}/Q_k). \tag{13}$$

We denote the set of coefficients with visual capacity for watermark embedding (i.e. capacity greater than zero) as

$$CV = \{c_1, ..., c_n\} = \{X_{i,j,k} | s_{i,j,k} \neq 0, \forall\, i, j, k\}. \tag{14}$$

Inserting the watermark in all of the coefficients in CV creates visible artifacts. The algorithms in [57, 76] only embed the watermark in those coefficients in CV that are greater than their corresponding visibility thresholds, $m_{i,j,k}$. However, this significantly limits the number of watermarked coefficients. If more coefficients need to be watermarked in the video frame, the parameters in the visual model that define the impact of masking need to be increased (a_T and $w_{i,j}$)[72]. The danger is that this may assume a greater benefit from masking than is actually available, resulting in noticeable visual artifacts. Furthermore, an adversary can more easily determine the locations of the watermarked coefficients, making the algorithm less robust to attacks.

A coefficient-selection algorithm chooses a subset of the coefficients in CV. A secret key controls the coefficient-selection process. The algorithm generates a palette that contains the actual locations of the watermarked coefficients. The owner can keep this palette for watermark detection or can compute the palette again using the watermarked video. This palette can be considered as an automatically generated confirmation number or password. Since the coefficient-selection algorithm is controlled by a key, the attacker does not know the actual location of watermarked coefficients in CV. To be confident of eliminating the watermark, an attacker needs to modify most of the coefficients in CV, which creates visible artifacts.

We designed the coefficient-selection algorithm to spread the watermark over frequencies and blocks. For each 4×4 block, each coefficient is ranked by a key. However, to spread the watermark over frequencies, we give a higher ranking for watermark embedding to those frequencies with the fewest coefficients in CV more frequently. We embed the watermark only in those coefficients whose magnitude is greater than a threshold, T_{coef}. The algorithm spreads the watermark over blocks by limiting the number of watermarked coefficients, T_{block}, that can be embedded in each block. Therefore, the algorithm embeds the watermark in the T_{block} highest ranking coefficients in CV that are greater than T_{coef}. One advantage of this strategy is that we can easily increase the watermark payload by increasing T_{block} or decreasing T_{coef}. Our experiments show that moderate relaxation of these thresholds increases the number of watermarked coefficients without impairing the visual quality, because the error pooling effect is limited. Furthermore, T_{block}, and T_{coef} can be adaptively adjusted to control the number of watermarked coefficients in a frame.

3.3 Watermark Embedding

We embed the watermark in the quantized DCT residuals of I-frames. Thus, only entropy decoding is required to embed the watermark, and the watermark embedding algorithm has low computational complexity. We use a bipolar watermark $W \in \{-1, 1\}$ with mean zero and variance one.

After any simple attack applied to the decoded video followed by reencoding, the residuals will change because the I-macroblock prediction modes will change. However, the linearity property of the DCT guarantees that the watermark is still present in the decoded video sequence, and we can still detect it with high probability. In the following, we denote the original pixel values by i_{ijk}, the prediction by p_{ijk}, the residual by r_{ijk}, and their corresponding DCTs by I_{ijk}, P_{ijk}, and R_{ijk}. Assume s is a DCT coefficient, then, $\tilde{s}$, s', and $\hat{s}$ represent the quantized, watermarked, and attacked coefficient, respectively. The addition of the prediction and residual is equal to the original pixel. This can be written as

$$i_{ijk} = p_{ijk} + r_{ijk}. \tag{15}$$

By the linearity property of the DCT,

$$I_{ijk} = P_{ijk} + R_{ijk}. \tag{16}$$

The watermark is inserted onto the quantized DCT residual as

$$\tilde{R}'_{ijk} = \tilde{R}_{ijk} + W_{ijk}, \tag{17}$$

and

$$R'_{ijk} = (\tilde{R}_{ijk} + W_{ijk}) \times Q_k. \tag{18}$$

Thus,

$$I'_{ijk} = I_{ijk} + W_{ijk}Q_k, \tag{19}$$

and the addition of the watermark to the quantized DCT residuals is the same as the addition of the watermark times the quantization step size to the original DCT coefficients. When common signal processing operations or watermarking attacks on the video change the prediction mode of the block, the residual and prediction will change to $\hat{r}_{ijk} \neq r'_{ijk}$, and $\hat{p}_{ijk} \neq p_{ijk}$. However, if the video quality is still acceptable, then $\hat{r}_{ijk} + \hat{p}_{ijk} = \hat{i}_{ijk} \approx i'_{ijk}$, and the watermark can still be extracted from the decoded video sequence.

3.4 Two Compressed-Domain Video Watermark Embedding Scenarios

When embedding a watermark in compressed video, there are two different possible scenarios. In the following two subsections, we show the structure of watermark embedding using the human visual model for these two scenarios and discuss their differences.

3.4.1 Watermark Embedding in the Encoder

In one scenario, the watermark is inserted in the encoder. This scenario is shown in Figure 9. If we ignore the top path in this block diagram, this is the structure of an H.264 encoder. Each macroblock of the current frame is predicted in either intra or inter prediction mode. The difference between the current macroblock and the prediction signal is the residual. The residuals are transformed, quantized, reordered and entropy coded and finally written to the bitstream. There is a backward path in the encoder that reconstructs the current frame. The perceptual model finds the location of coefficients with watermarking capacity, CV, using the original video frame. The coefficient selection algorithm selects a subset of coefficients from the coefficients with watermarking capacity and the watermark is added to the quantized DCT residuals at those locations.

In this case, the error induced by watermarking will be corrected in future predictions and will not propagate within I-frames or to P-frames. Consequently, more coefficients can be watermarked in the compressed video while maintaining high perceptual quality. However, there will be an increase in the bit rate of the macroblocks that are predicted from the watermarked macroblocks.

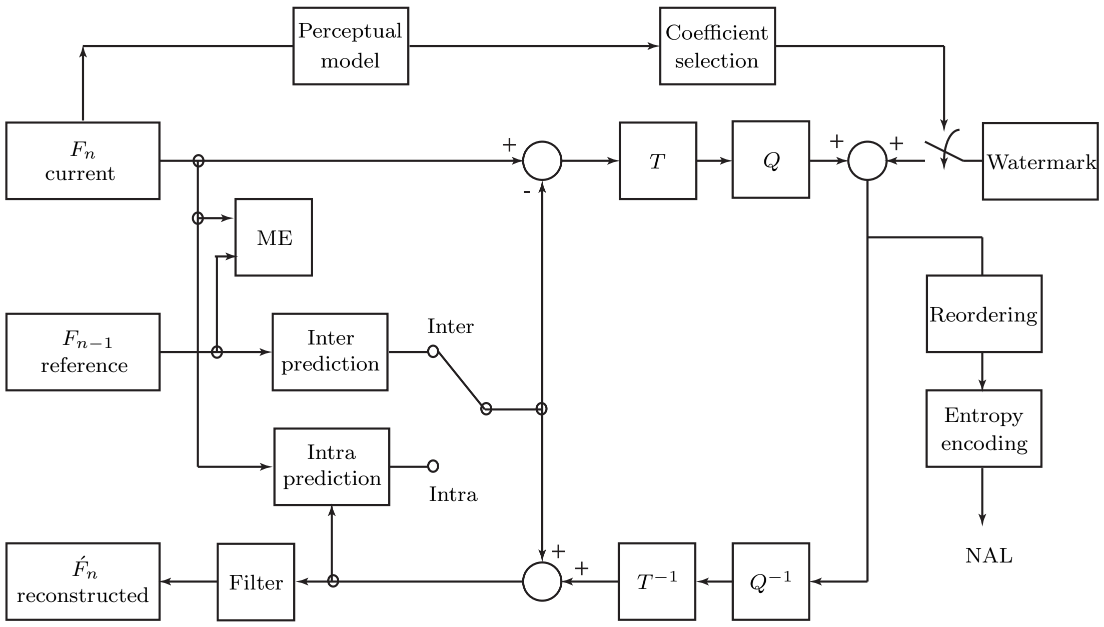


Figure 9: Perceptual watermark embedding in the encoder.

3.4.2 Watermark Embedding in the Bitstream

In the other scenario, the watermark is embedded into the bitstream. This scenario is shown in Figure 10. In this case, error propagation makes maintaining high visual quality a bigger problem than the increase in video bit rate. Since H.264 uses intra-prediction in I-frames, the error propagates both within I-frames and to P-frames, with the error propagation in I-frames more severe than in P-frames. One watermarked coefficient in an I-frame is likely to affect only one pixel in each of the P-frames predicted from that frame. However, one watermarked coefficient can change a whole 4×4 or 16×16 block predicted from the block that has the watermarked coefficient in an I-frame. To have acceptable visual quality, a drift compensation signal needs to be added in the decoder to compensate for the error as suggested in [17].

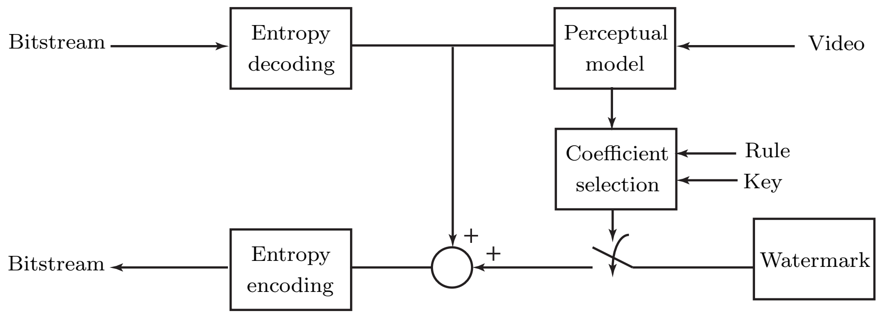


Figure 10: Perceptual watermark embedding in the bitstream.

3.5 Exploring the Visibility of Propagated Artifacts in P-frames for Bitstream Watermarking

When the watermark is embedded in the bitstream, the watermark signal in I-frames may propagate within I-frames and to the following P- and B-frames and produce undesirable artifacts. Hartung *et al.* used drift compensation to compensate for propagated errors in P- and B-frames [17]. However, the computational complexity of this solution limits its applicability. Nonetheless, it is important to embed the watermark so that the propagated artifacts in P- and B-frames are as small as possible. In this section, we propose using drift compensation as suggested in [17] to compensate for the error in I-frames but not P-frames. The error propagation in I-frames is more severe than P-frames as mentioned in the Subsection 3.4.2. Since the frequency of I-frames is low in the video sequence, the computational complexity cost will be smaller than using drift compensation for the whole video sequence.

Our intuition suggests that the propagated artifacts in P-frames are often more noticeable in moving areas because watermark errors can move from locations with high spatial masking thresholds in I-frames to locations with lower spatial masking thresholds in P- and B-frames. Furthermore, moving errors are usually more objectionable than static errors, and the human eye naturally pays more attention to moving areas. On the other hand, temporal masking suggests that the moving areas in a frame have higher masking thresholds. Therefore, in this section we find the moving areas in the video frame to explore the visibility of propagated artifacts in those areas. We make the number of watermarked coefficients in those areas adaptive by changing T_{block} and T_{coef}, and we observe the visual quality of the video. Although the information inferred from motion estimation does not represent true motion, it still carries useful information for finding an estimate of motion characteristics. In the following two subsections, we show how to find the moving areas using motion intensity in a frame and a motion history image (MHI) inferred from motion estimation.

3.5.1 Motion Intensity

P-frames in H.264 contain intra-coded, inter-coded and copy macroblocks. The copy mode indicates a macroblock is a direct copy of the macroblock at the same location in the previous reference frame. In inter-coded macroblocks, each partition is predicted from an area of the same size in a reference frame. Inter-coded macroblocks in a P-frame are predicted from a previously coded frame using motion compensation. In intra-coded macroblocks each 16×16 or 4×4 luma region and each 8×8 chroma region is predicted from previously-coded samples in the same slice. Macroblocks

Table 3: Number of macroblocks coded in each mode for the first P-frame of six standard video sequences.

Sequence	Copy	Intra	Inter
CLAIRE	71	0	28
CARPHONE	30	5	69
MOBILE	7	0	92
MOTHER	70	0	29
SALESMAN	85	0	14
TABLE	66	0	33

encoded in intra-coded mode often represent uncovered background. The number of macroblocks of each prediction type represents the amount of motion versus static background in each video frame. For example, video sequences with a large fraction of static background can be expected to have a larger number of macroblocks coded in copy mode. We denote the number of macroblocks coded in copy mode as N_{copy}. Table 3 shows the number of macroblocks coded in each mode for the first P-frame of six standard video sequences in QCIF (176×144) format. In this resolution, there are 99 macroblocks per frame.

In video sequences with large static backgrounds such as CLAIRE, MOTHER, TABLE and SALESMAN, a large number of macroblocks are coded in copy mode. In video sequences where most of the frame is moving, such as MOBILE, few macroblocks are coded in copy mode. The motion of the video sequence CARPHONE lies between these extremes.

3.5.2 Spatial Motion Distribution

The motion history image (MHI) has been used in [31] to control motion estimation. Each pixel (i, j) in an MHI corresponds to the spatial $(i, j)^{th}$ block in a sequence. The pixel intensity describes how long since there has been motion detected at that location. Let $I_f(i, j)$ be the pixel intensity at time index f, with $I_0(i, j) = 0$. At each frame, the pixel intensity is updated as follows:

$$I_f(i,j) = \begin{cases} I_{f-1}(i,j) + 1 & |mv_x^f(i,j)| + |mv_y^f(i,j)| = 0 \\ 0 & |mv_x^f(i,j)| + |mv_y^f(i,j)| \neq 0 \end{cases} \tag{20}$$

where $(mv_x^f(i,j), mv_y^f(i,j))$ is the motion vector of block (i,j) at frame f. The MHI can be considered as a histogram of non-moving regions.

We use the MHI to find the motion history in I-frames from the motion estimation information in the previous GOP P-frames. The intra period is set to four, and we modify equation (20) such that the blocks with $|mv_x^f(i,j)| + |mv_y^f(i,j)| < T_{mv}$ are considered static. The algorithm adaptively calculates T_{mv} based on the motion intensity of the current GOP as follows:

$$T_{mv} = \begin{cases} 3 & N_{copy} \geq 70, \\ 3 + 3(7 - floor(N_{copy}/10)) & N_{copy} < 70. \end{cases} \tag{21}$$

This equation, found experimentally, provides a tradeoff between finding enough moving and static blocks. We assume that copy-mode macroblocks have nearly zero motion vectors and that intra-coded macroblocks have large motion vectors. The first and second I-frames of two standard videos with their corresponding motion vectors from the P-frame before the second I-frame and their MHI are shown in Figure 11. For display purposes, each pixel of the MHI is depicted as one block. The pixel intensity is normalized between 0 and 255. The black areas in the MHI indicate there has been motion in those areas in the previous P-frame and they correspond to the non-white areas in the image that is constructed from the magnitude of motion vectors of 4×4 blocks, what we call MV. We observe that the motion history image not only contains motion history information, but it also gives a good estimate of the spatial motion distribution.

3.6 Simulation Results

We implemented our proposed watermarking algorithm in the H.264 reference software version JM10.2 [1]. We considered the scenario that the watermark was embedded in the encoder. To compare the perceptual quality of our proposed watermarking algorithm with the algorithm in [76], an I-frame and the following P-frame from the standard video sequence CARPHONE (QCIF, 176×144) are shown in Figure 12. The frames on the top are watermarked using our algorithm, and the frames on the bottom are watermarked using the approach in [76]. Note that only the I-frame is watermarked, but adding a watermark to an I-frame will affect its dependent P-frames. This figure shows that the perceptual qualities of the watermarked frames from the two algorithms are comparable. However, there are 929 watermarked coefficients in the I-frame watermarked with our algorithm whereas the I-frame watermarked with the algorithm in [76] has only 642 watermarked coefficients.

We used six standard QCIF video sequences (176×144) at the rate of 30 frames per second for our simulation. We choose $T_{block} = 2$ and $T_{coef} = 10$. Table 4 shows the percentage of the watermarked coefficients from the set of coefficients with visual watermarking capacity, CV, and the average number of watermarked coefficients in each I-frame for each video sequence for our algorithm versus the algorithm in [76]. The results show that our algorithm increases the number of watermarked coefficients for all video sequences except the video sequences MOBILE and TABLE. These video sequences are highly textured. Thus, they have a large number of DCT coefficients with watermarking capacity. However, our algorithm limits the number of watermarked coefficients in each block to $T_{block} = 2$, which decreases the number of watermarked coefficients in those video sequences. The performance of our detection algorithm proposed in Chapter 4 depends on the number of watermarked coefficients in each interval. Since these video sequences already have many watermarked coefficients in each frame, this does not influence the performance of our detection algorithm. It

Table 4: Percentage of watermarked coefficients from the set of coefficients with visual watermarking capacity and the average number of watermarked coefficients in each I-frame for six standard video sequence.

	Percentage of watermarked coefficients from CV		Average number of watermarked coefficients in an I-frame	
Sequence	Our alg.	Alg. in [76]	Our alg.	Alg. in [76]
CARPHONE	19.0%	8.8%	891	609
CLAIRE	6.6%	5.0%	450	346
MOBILE	19.0%	22.4%	2291	2699
MOTHER	7.6%	3.7%	630	309
SALESMAN	20.0%	13.3%	953	626
TABLE	8.0%	8.8%	810	897

is more important to increase the number of watermarked coefficients in those video sequences that have few DCT coefficients with watermarking capacity. On average, watermark embedding using our algorithm increases the bit rate of the video by about 5.6% versus 4.3% using the algorithm in [76]. Since these algorithms use human visual models, the PSNR is not an appropriate metric to compare the visual quality. However, readers might find it useful to know that the PSNR of the watermarked video decreases 0.58 dB compared to the compressed (but unwatermarked) video for our algorithm versus 0.48 dB for the algorithm in [76].

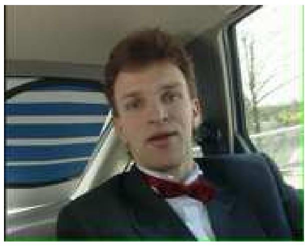
To explore the advantage of using motion characteristic to reduce the visibility of propagated artifacts in P-frames when the watermark is embedded in the bitstream, we implemented our proposed watermarking algorithm in the H.264 bitstream using the reference software version JM10.1 [1] with an intra period of four. To distinguish between watermarking artifacts from intra-prediction in I-frames and motion estimation in P- and B-frames, we prevented the H.264 encoder from using intra-prediction in I-frames. We found the moving areas of the video frame using the motion intensity and motion history images. To explore the advantage of using the motion activity model, we ran the following experiment. We set $T_{coef} = 10$ and $T_{block} = 2$

for static areas, and we changed T_{coef} and T_{block} in the moving areas. Our simulation results showed that no significant change in visual quality was observed when T_{block} was changed from zero to five and T_{coef} was changed from one to 10. Thus, exploiting motion characteristics did not have any advantage in terms of improving the perceptual quality of the video in this case.

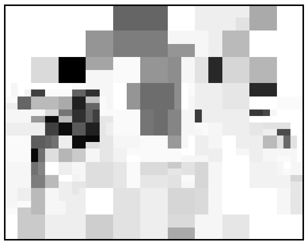
In Chapter 4, we develop a watermark detection algorithm with controllable performance and we use it to detect the watermark embedded in the compressed video using the algorithm presented in this chapter. Our simulation results show that we achieve the desired detection performance in Monte Carlo trials. The simulation results also show that our proposed watermarking scheme is robust to 3×3 Gaussian filtering, 50% cropping, addition of white noise, $\mathcal{N}(0, 0.001)$, and a trivial deliberate attack.

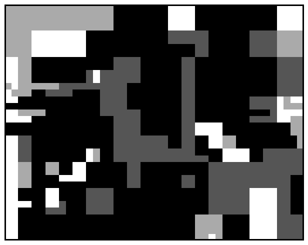
3.7 Conclusion

In this chapter, we proposed a watermarking algorithm for H.264 that is robust to common signal processing attacks. We achieved this goal by employing Watson's human visual model adapted for a 4x4 DCT block to obtain a larger payload and a greater robustness while minimizing visual distortion. We used a key-dependent algorithm to select a subset of the coefficients with visual watermarking capacity for watermark embedding to obtain robustness to malicious attacks. Furthermore, we spread the watermark over frequencies and within blocks to avoid error pooling. Our simulation results show that we increased the payload and robustness without a noticeable change in perceptual quality by reducing this effect. We embedded the watermark in the coded residuals to avoid decompressing the video. However, we detected the watermark from the decoded video sequence in order to make the algorithm robust to intra-prediction mode changes. The result of this work appears in [39, 41].


(a) First I-frame

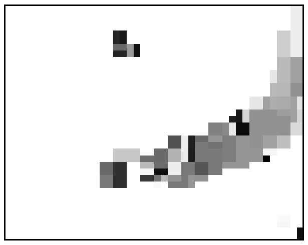
(b) Second I-frame


(c) Previous P-frame MV


(d) CARPHONE's MHI

(e) First I-frame

(f) Second I-frame


(g) Previous P-frame MV

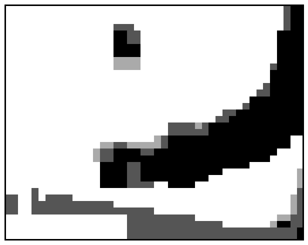
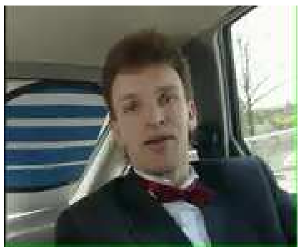

(h) TABLE's MHI

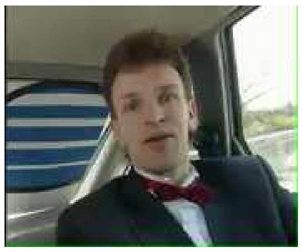
Figure 11: CARPHONE and TABLE video sequences with their corresponding motion vector (MV) and motion history images (MHI).

(c) Our alg.'s watermarked I-frame

(d) Our alg.'s Watermarked P-frame



(c) Watermarked I-frame from [76]



(d) Watermarked P-frame from [76]

Figure 12: Comparison of the visual quality of our perceptual watermark embedding algorithm with the algorithm in [76].

CHAPTER IV

LOCATION-AWARE DETECTION (LAD)

The performance of any watermarking scheme relies heavily on the design of the watermark detector. Zeng *et al.* argued that for the particular application of resolving rightful ownership using invisible watermarks, it is crucial that the original image not be directly involved in the watermark detection process [80]. The true owner should be able to detect the watermark without using a second image, since its authenticity is also questionable. Hernandez *et al.* presented a watermark detection algorithm where the embedding domain is the DCT coefficients except the dc term [19]. Their algorithm assumes a generalized Gaussian distribution for the ac coefficients of the DCT and an additive embedding rule. In [35], Nikolaidis *et al.* considered watermarking in the DCT and DWT domains and the same pdf assumption for the coefficients as in [19]. The authors employ a Rao test that is equivalent to a generalized likelihood ratio test. The resulting detector is asymptotically optimal, meaning that it is optimal under the assumption of a large data record. Huang *et al.* presented a new detection structure for transform domain additive watermarks based on Huber's robust hypothesis testing theory [20]. The statistical behaviors of the image subband coefficients are modeled by a contaminated generalized Gaussian, which tries to capture a small deviation of the actual situation from the idealized generalized Gaussian. The performance of these algorithms has been tested on different images.

In this chapter, we build a theoretical framework for watermark detection using a likelihood ratio test, and we use this framework to obtain video watermark detection with controllable performance [41, 42]. We show that performance of our video watermark detector depends upon three parameters: the average of the squares of the H.264 quantization step sizes of watermarked DCT coefficients, the standard deviation of watermarked DCT coefficients, and the number of watermarked DCT coefficients, N_w, over which the detector response is computed. We cannot control the standard deviation of the DCT coefficients or the quantization step size, but, when detecting watermarks in video, we can control the number of watermarked coefficients used to compute the detector response. This is not the case with images, since there is a limited number of coefficients that can be watermarked in each image before the watermark is visible. Therefore, our video watermark detection algorithm calculates N_w to obtain the desired probability of a detection, P_D, for a given probability of a false alarm, P_F. Our simulation results show that the theoretically chosen value for N_w does lead to the desired values of P_D and P_F in Monte Carlo trials.

4.1 Theoretical Framework

Watermark detection is a classical detection problem [71] where one hypothesis states that the watermark is present and the other states that the watermark is not present. Detecting the watermark requires choosing between the two hypotheses. The observations under the two hypotheses are as follows:

$$\begin{aligned} &H_0 : y_\ell = I_\ell && \ell \in CW, \\ &H_1 : y_\ell = I_\ell + W_\ell Q_\ell && \ell \in CW, \end{aligned} \tag{22}$$

where CW is the set of watermarked coefficients, I_ℓ is the selected DCT coefficient of the video frame, Q_ℓ is the H.264 quantization step size selected by the video encoder for that coefficient and W_ℓ is chosen from a bipolar watermark sequence with mean zero and variance one. The index ℓ denotes the ℓ^{th} watermark bit or the ℓ^{th} DCT coefficient. It has been shown that the ac coefficients of the DCT are well modeled by a generalized Gaussian distribution [6, 22] with mean zero, which can be written as

$$p_{I_\ell}(I) = ae^{-|b(I)|^c}, \tag{23}$$

where a and b are defined as

$$a = \frac{bc}{2\Gamma(\frac{1}{c})}, \tag{24}$$

$$b = \frac{1}{\sigma_w}\sqrt{\frac{\Gamma(\frac{3}{c})}{\Gamma(\frac{1}{c})}}, \tag{25}$$

and $\Gamma(.)$ is the gamma function. A value of $c = 0.8$ models the ac coefficients reasonably well. The value of $c = 2$ results in a normal Gaussian distribution. For simplicity, we assume that $c = 2$ and show that it results in low complexity watermark detection algorithm with controllable detection performance. The performance and results of the detector when $c \neq 2$ are still valid, but, in this case the detector is suboptimal. The ac coefficients probability distribution with $c = 2$ is

$$p_{I_\ell}(I) = \frac{1}{\sigma_w\sqrt{2\pi}}e^{-\frac{(I)^2}{2\sigma_w^2}}. \tag{26}$$

The optimal detector compares the likelihood ratio to a threshold

$$\frac{p_{y|H_1}(Y|H_1)}{p_{y|H_0}(Y|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \eta \tag{27}$$

where η controls the tradeoff between missed detections and false alarms [71]. Assuming that the watermarked DCT coefficients have Gaussian distribution, and they are statistically independent and substituting the joint probability density into the likelihood ratio test in (27) gives

$$\frac{\prod_{\ell=1}^{N_w}\frac{1}{\sigma_w\sqrt{2\pi}}e^{-\frac{(Y_\ell - W_\ell Q_\ell)^2}{2\sigma_w^2}}}{\prod_{\ell=1}^{N_w}\frac{1}{\sigma_w\sqrt{2\pi}}e^{-\frac{(Y_\ell)^2}{2\sigma_w^2}}} \underset{H_0}{\overset{H_1}{\gtrless}} \eta. \tag{28}$$

Algebraic simplification reduces this to the equivalent test

$$Y = \sum_{\ell=1}^{N_w} Y_\ell W_\ell Q_\ell \underset{H_0}{\overset{H_1}{\gtrless}} \sigma_w^2 \ln\eta + \frac{N_w \bar{Q}_w^2}{2}, \tag{29}$$

where N_w is the number of DCT coefficients from CW and $\bar{Q}_w^2 = (1/N_w)\sum_{\ell=1}^{N_w} Q_\ell^2$. We see that we can detect the watermark by multiplying the DCT coefficients in CW of the decoded frame, Y_ℓ, by the original watermark bits, $W_\ell Q_\ell$, calculating the sum of those terms, and comparing the result with a threshold.

Assuming that the DCT coefficients are independent and applying the central limit theorem, Y is $\mathcal{N}(0, N_w\bar{Q}_w^2\sigma_w^2)$ under H_0, and Y is $\mathcal{N}(N_w\bar{Q}_w^2, N_w\bar{Q}_w^2\sigma_w^2)$ under H_1, where σ_w is the standard deviation of DCT coefficients in CW. Multiplying (29) by $1/(\sigma_w\sqrt{N_w\bar{Q}_w^2})$ to normalize the Gaussian distribution, we have

$$\psi = \frac{1}{\sigma_w\sqrt{N_w\bar{Q}_w^2}}\sum_{\ell=1}^{N_w} Y_\ell W_\ell Q_\ell \underset{H_0}{\overset{H_1}{\gtrless}} \frac{\sigma_w \ln\eta}{\sqrt{N_w\bar{Q}_w^2}} + \frac{\sqrt{N_w\bar{Q}_w^2}}{2\sigma_w}, \tag{30}$$

where the threshold, T, is

$$T = \frac{\sigma_w \ln\eta}{\sqrt{N_w\bar{Q}_w^2}} + \frac{\sqrt{N_w\bar{Q}_w^2}}{2\sigma_w}. \tag{31}$$

Then, ψ is $\mathcal{N}(0,1)$ under H_0 and is $\mathcal{N}(\sqrt{N_w\bar{Q}_w^2}/\sigma_w, 1)$ under H_1. These probability densities are shown in Figure 13. The distance between the means of the two densities is $d \triangleq (\sqrt{N_w\bar{Q}_w^2})/\sigma_w$.

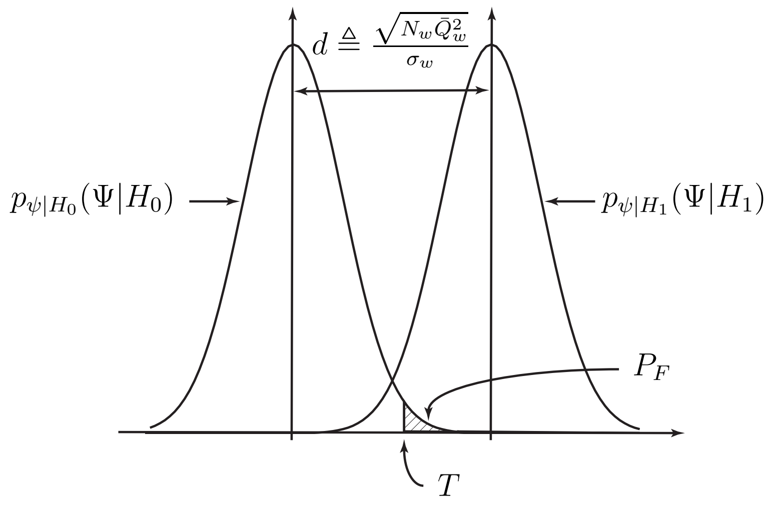


Figure 13: Probability densities $p_{\psi|H_0}(\Psi|H_0)$ and $p_{\psi|H_1}(\Psi|H_1)$ for a location-aware detector.

To evaluate the performance of the watermark detector, we compute the probability of a detection, P_D, and the probability of a false alarm, P_F. P_D and P_F are as follows:

$$P_D = \int_T^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-d)^2}{2}} dx = \mathrm{erfc}(T-d), \tag{32}$$

$$P_F = \int_T^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \mathrm{erfc}(T) \tag{33}$$

To achieve the specified value of P_D and P_F, the detector selects the threshold to agree with the value of P_F and then selects d to achieve the target values of P_D. Recall that d is a function of three parameters and is defined as

$$d \triangleq \frac{\sqrt{N_w \bar{Q}_w^2}}{\sigma_w}. \tag{34}$$

$\bar{Q}_w^2$ is the average of the squares of the H.264 quantization step sizes of the DCT coefficients in CW and is chosen by the encoder. σ_w is the standard deviation of the DCT coefficients in CW and is a property of the video. Therefore, the watermark detector cannot change either of these two parameters. However, the third parameter, N_w, is the number of DCT coefficients in CW used to compute the detector response, and it can be chosen by the detection algorithm to obtain the desired value of d. The detector finds the value of N_w by solving (34) for it. The detector then computes the detector response, ψ, over N_w coefficients in CW. The stream of N_w watermarked DCT coefficients may extend across several I-frames or may be contained within a fraction of an I-frame. Note that if $Q_\ell = Q$ is fixed for all the watermarked DCT coefficients, then, (34) simplifies to

$$d \triangleq \frac{\sqrt{N_w} Q}{\sigma_w}. \tag{35}$$

Our watermark detection scheme has several advantages. First, the error rate of the detector can be maintained regardless of the video sequence given that the video is long and the detector response latency can be arbitrary. The detector response, ψ, has the same probability distribution regardless of the video sequence. However, based upon N_w and the number of watermarked DCT coefficients in each I-frame, the number of I-frames needed to compute ψ varies. This means that the detector will produce results more frequently for some videos than others. We believe that this is acceptable since nearly every video should have a sufficient number of watermarked DCT coefficients to produce detector responses at an acceptable rate. Another advantage is that the responsibility of choosing the value of N_w lies completely with the detector and does not place any burden on the watermark embedding system. For example, if the watermark detector notices that the video sequence has been attacked to remove the watermark, it can increase the value of N_w to obtain more reliable detector response. Notice that we are taking advantage of the large amount of data in video sequences compared to images to obtain more robust watermark detection. Another advantage is that computing the detector response, ψ, has low computational complexity. Recall that ψ is defined as

$$\psi = \frac{1}{\sigma_w \sqrt{N_w \bar{Q}_w^2}} \sum_{\ell=1}^{N_w} Y_\ell W_\ell Q_\ell. \tag{36}$$

The watermark sequence is a bipolar sequence of $\{-1, 1\}$, and usually Q_ℓ is constant within a subset of DCT coefficients that are used to compute one detector response. Thus, computing the detector response requires only the addition or subtraction of the DCT coefficients with few multiplications. However, choosing N_w requires computing the standard deviation of coefficients in CW, which has high computational complexity, but this standard deviation varies over a small range. Thus, if the computational complexity of computing σ_w for the video frames is undesirable, one can always assume an upper limit on the value of σ_w and set N_w accordingly.

4.2 Simulation Results

To test the performance of our watermark detection scheme, we implemented our watermark embedding scheme proposed in Chapter 3 in the H.264 reference software version JM10.2 [1]. We considered the scenario that the watermark was embedded in the encoder, and we used six standard video sequences in the QCIF (176×144) format at the rate of 30 frames per second. To compare the experimental results with the theoretical framework derived in the previous section, a large number of watermarked coefficients are required to compute the detector response many times. Thus, we coded and watermarked I-frames of every video sequence 80 times by an H.264 encoder with an intra period of one (group of pictures: I B I). Note that the more detector responses we have, the more smoothly we can estimate their distribution. The H.264 encoder used a fixed quantization step size $Q = 16$ for I-frames.

The goal of our first experiment is to obtain $P_D = 0.99$ and $P_F = 0.01$. Solving (32) and (33) analytically, these probabilities can be achieved for $T = 2.325$ and $d = 4.65$. Comparing the watermarked coefficients from the decoded video with their values before the watermark was added, we notice that the difference is not exactly equal to the quantization step size, because the encoding process is lossy. On average, the difference between the watermarked coefficients and their values before watermarking is $\hat{Q}$ instead of Q, where $\hat{Q}$ is generally smaller than Q, but close to it. Therefore, we have to solve

$$d \triangleq \frac{\sqrt{N_w}\hat{Q}}{\sigma_w}, \tag{37}$$

to find N_w. This equation suggests that the smaller the values of the quantization parameter are, the larger N_w must be to obtain the desired performance. Also the larger the value of σ_w, the larger N_w must be to obtain the desired performance.

Table 5 shows the average value of $\hat{Q}$ and σ_w calculated over 80 watermarked sequences for each video and the corresponding N_w to obtain $d = 4.65$. We detect the watermark by computing the detector response over N_w watermarked coefficients for each video. The threshold is set at $T = d/2 = 2.325$. We calculate the probability of a detection and probability of a false alarm based on this threshold. P_D and P_F and the mean value of the detector response, m_ψ, obtained from our experiments are also shown in Table 5. Our results show that P_D is close to 0.99, P_F is close to 0.01, and m_ψ is close to $d = 4.65$ for all the video sequences. In Figure 14, we

Table 5: Experimental results when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-aware detector.

Sequence	$\hat{Q}$	σ_w	N_w	m_ψ	P_D	P_F
CARPHONE	14.40	59.17	365	4.68	0.9898	0.0108
CLAIRE	13.52	81.73	790	4.67	0.9905	0.0114
MOBILE	15.37	51.86	246	4.71	0.9917	0.0121
MOTHER	12.91	63.65	526	4.64	0.9877	0.0140
SALESMAN	14.33	54.60	314	4.67	0.9913	0.0109
TABLE	14.95	59.45	342	4.70	0.9920	0.0107

plot the probability distribution of the detector response under H_0 and H_1 for the video sequence CARPHONE. We have a total of 31978 detector responses from coding this video sequence 80 times. The symbols $\circ$ and $\diamond$ reflect the number of detector responses in the intervals centered around them. We have scaled the number of detector responses in each interval so that their largest value has the same value as the peak of a Gaussian distribution with variance one. This figure shows that the experimentally determined $p(\Psi|H_0)$ and $p(\Psi|H_1)$ approximate a normal Gaussian distribution with a variance of one. This justifies our assumption that the detector response has a normal Gaussian distribution.

In the next experiment, the target is to obtain $P_D = 0.999$ and $P_F = 0.001$. Solving equations (32) and (33) analytically, these probabilities can be achieved with $T = 3.09$ and $d = 6.18$. Table 6 shows the average values of $\hat{Q}$ and σ_w and the corresponding N_w to obtain $d = 6.18$. P_D and P_F and the mean of the detector response, m_ψ, obtained from our experiments are also shown in Table 6. Again our results show that P_D is close to 0.999 and P_F is close to 0.001 for all the video sequences. In Figure 15, we plotted the probability distribution of the detector response under H_0 and H_1 for the video sequence CARPHONE. This figure shows that $p(\Psi|H_0)$ and $p(\Psi|H_1)$ are further apart than $p(\Psi|H_0)$ and $p(\Psi|H_1)$ in Figure 14.

In the next experiment, the target is to obtain $P_D = 0.99$ and $P_F = 0.001$. From

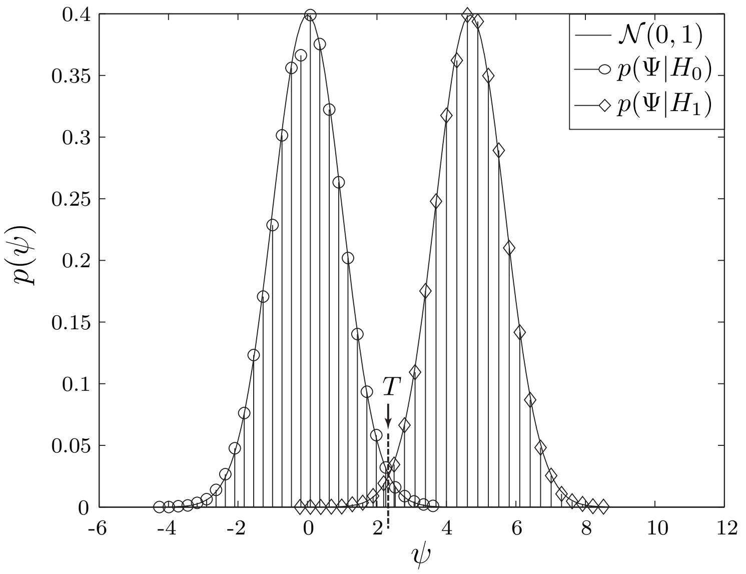


Figure 14: Detector response probability distribution of a location-aware detector to achieve $P_D = 0.99$ and $P_F = 0.01$ for CARPHONE video sequence.

Table 6: Experimental results when the target is to achieve $P_D = 0.999$ and $P_F = 0.001$ for a location-aware detector.

Sequence	$\hat{Q}$	σ_w	N_w	m_ψ	P_D	P_F
CARPHONE	14.40	59.17	645	6.21	0.9992	0.0009
CLAIRE	13.52	81.73	1395	6.20	0.9995	0.0014
MOBILE	15.37	51.86	435	6.25	0.9990	0.0014
MOTHER	12.91	63.65	928	6.16	0.9991	0.0015
SALESMAN	14.33	54.60	554	6.20	0.9992	0.0009
TABLE	14.95	59.45	604	6.23	0.9993	0.0016

(33), we find that $T = 3.09$ achieves $P_F = 0.001$. Then, from (32), we find that $d = 5.416$ achieves $P_D = 0.99$. Finally, we solve (37) to obtain the required N_w for each video sequence. Table 7 gives P_D, P_F and the mean of the detector response, m_ψ, for each video sequence. Figure 16 shows $p(\Psi|H_0)$ and $p(\Psi|H_1)$, and the threshold T for the video sequence CARPHONE for this case. Since the probability of a false alarm is smaller than the probability of a missed detection, the threshold is to the right of where $p(\Psi|H_0)$ and $p(\Psi|H_1)$ intersect.

Finally, we look at the effect of different attacks on detection performance. We first

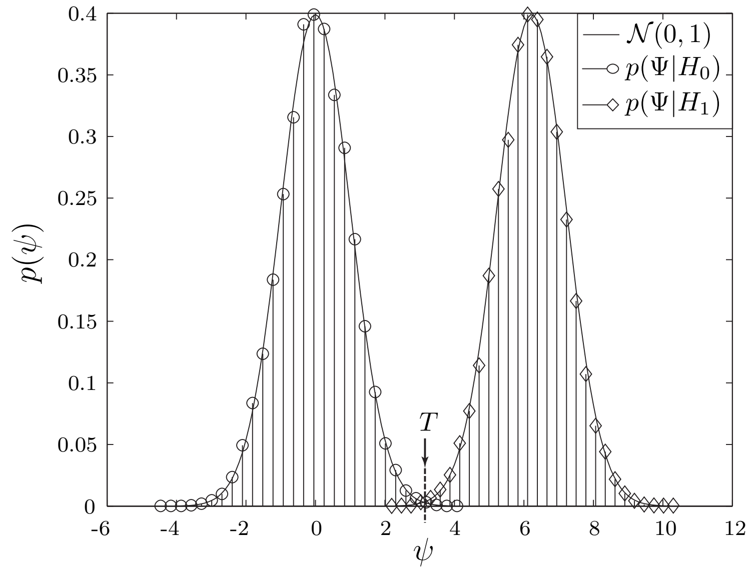


Figure 15: Detector response probability distribution of a location-aware detector to achieve $P_D = 0.999$ and $P_F = 0.001$ for CARPHONE video sequence.

Table 7: Experimental results when the target is to achieve $P_D = 0.99$ and $P_F = 0.001$ for a location-aware detector.

Sequence	$\hat{Q}$	σ_w	N_w	m_ψ	P_D	P_F
CARPHONE	14.40	59.17	495	5.44	0.9904	0.0012
CLAIRE	13.52	81.73	1071	5.43	0.9895	0.0009
MOBILE	15.37	51.86	334	5.48	0.9921	0.0012
MOTHER	12.91	63.65	713	5.40	0.9882	0.0012
SALESMAN	14.33	54.60	426	5.44	0.9916	0.0009
TABLE	14.95	59.45	464	5.46	0.9924	0.0013

consider a 3×3 Gaussian filtering attack. We choose N_w and the threshold, $T = 3.09$, as in Table 6 to approximate $P_D = 0.999$ and $P_F = 0.001$ without any attack. Table 8 gives $\hat{Q}$, σ_w, m_ψ, P_D and P_F after the Gaussian filtering attack. Comparing Table 8 with Table 6 shows that after the Gaussian filtering attack the variance of the video sequences remains approximately the same, but $\hat{Q}$ becomes significantly smaller. Thus, if we choose N_w as before, $p(\Psi|H_1)$ moves towards $p(\Psi|H_0)$ because the mean value of ψ under hypothesis H_1, m_ψ, becomes smaller. Since we set the threshold T as before, we still obtain the desired $P_F = 0.001$, however P_D is lower

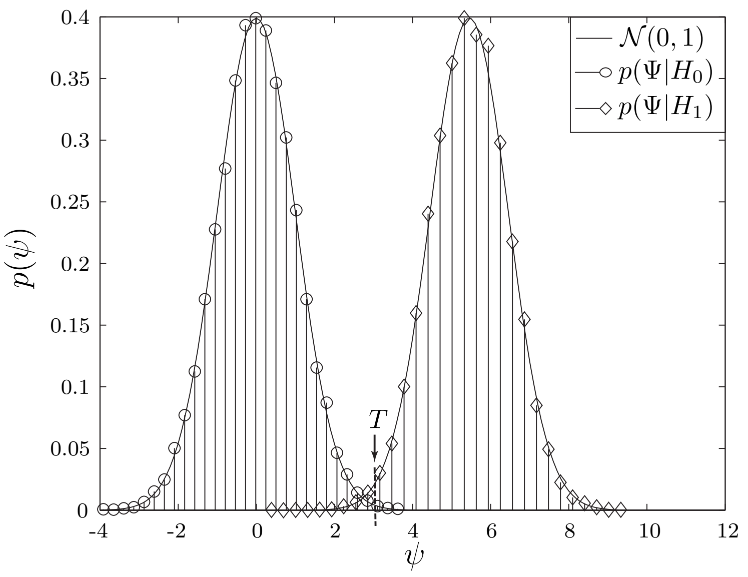


Figure 16: Detector response probability distribution of a location-aware detector to achieve $P_D = 0.99$ and $P_F = 0.001$ for CARPHONE video sequence.

Table 8: Experimental results after the 3×3 Gaussian filtering attack when the target is to achieve $P_D = 0.999$ and $P_F = 0.001$ for a location-aware detector.

Sequence	$\hat{Q}$	σ_w	N_w	m_ψ	P_D	P_F
CARPHONE	10.02	60.12	645	4.25	0.9118	0.0009
CLAIRE	9.56	81.01	1395	4.42	0.9300	0.0014
MOBILE	9.43	51.92	435	3.79	0.8300	0.0014
MOTHER	9.62	65.11	928	4.49	0.9400	0.0015
SALESMAN	10.08	54.57	554	4.36	0.9380	0.0009
TABLE	9.35	59.59	604	3.85	0.8500	0.0016

than 0.999. Comparing Table 8 with Table 5 shows that the m_ψ's obtained for each video sequence are similar. Thus, we can still achieve $P_D = 0.99$ and $P_F = 0.01$ by choosing T as we did for Table 5.

Increasing N_w further, we can still achieve $P_D = 0.999$, and $P_F = 0.001$. Suppose that after the Gaussian filtering attack, $\hat{Q}$ becomes as small as 9 and suppose that the variance of the video sequences remains the same. We use these values to calculate the required N_w for each video sequence. Table 9 shows the assumed $\hat{Q}$, the calculated value for N_w, and P_D and P_F after the 3×3 Gaussian filtering attack. Since we

Table 9: Experimental results after the 3×3 Gaussian filtering attack with new values of N_w when the target is to achieve $P_D = 0.999$ and $P_F = 0.001$ for a location-aware detector.

Sequence	$\hat{Q}$	σ_w	N_w	m_ψ	P_D	P_F
CARPHONE	9	59.17	1650	6.79	> 0.9998	0.0017
CLAIRE	9	81.73	3149	6.65	> 0.9992	0.0007
MOBILE	9	51.86	1268	6.47	> 0.9999	0.0011
MOTHER	9	63.65	1910	6.44	> 0.9997	0.0009
SALESMAN	9	54.60	1405	6.94	> 0.9999	0.0011
TABLE	9	59.45	1666	6.40	> 0.9998	0.0007

assumed that $\hat{Q}$ gets smaller than it actually does and we set the threshold as before, we always do better in detection than $P_D = 0.999$. We did not find any missed detections for any of the video sequences in our experiment. Therefore, the only statement that we can make is that the probability of missed detection, P_M, is smaller than one over the number of detector responses we computed, and P_D is greater than 1 minus this value.

We also consider a cropping attack. In our experiment, we crop each video frame to approximately 50% of its original size from the four sides. We assume that the detector can determine how the video is cropped by using either the original video sequence or synchronization templates. The simulation results show that the cropping attack does not affect the detection performance, however, it does affect the number of detector responses that can be extracted for each video sequence. Table 10 shows N_w and the number of QCIF (176×144) I-frames, F, required to compute the detector response for different detection performance scenarios and after Gaussian filtering and cropping attacks. Assume that video is displayed at the rate of 30 frames per second and an I-frame is sent once per second. Then, this table suggests that we will be able to extract one detector response in every F seconds. We see that the largest value of $F = 7.41$ happens for the CLAIRE video sequence after the Gaussian filtering attack. Note that the QCIF format (176×144) has one of the smallest resolutions;

Table 10: Number of I-frames required to compute the detector response for different detection performance scenarios and after the 3×3 Gaussian filtering and 50% cropping attacks for a location-aware detector.

	No attack $P_D = 0.99$ $P_F = 0.01$		No attack $P_D = 0.999$ $P_F = 0.001$		Filtering $P_D = 0.999$ $P_F = 0.001$		Cropping $P_D = 0.999$ $P_F = 0.001$	
Sequence	N_w	F	N_w	F	N_w	F	N_w	F
CARPHONE	365	0.46	645	0.82	1650	2.09	645	1.37
CLAIRE	790	1.85	1395	3.28	3149	7.41	1395	5.69
MOBILE	246	0.12	435	0.21	1268	0.61	435	0.37
MOTHER	526	0.95	928	1.64	1910	2.67	928	3.47
SALESMAN	314	0.39	554	0.70	1405	1.78	554	1.01
TABLE	342	0.46	604	0.82	1666	2.26	604	1.41

these results improve for higher resolution videos. Furthermore, we only embed the watermark in the luminance component of I-frames. The number of watermarked coefficients and subsequently the frequency of the detector response can be increased by embedding the watermark in P-frames and/or chroma components.

Next, we consider the effect of additive white noise. We add white noise of mean zero and variance 0.001 to each frame of the video sequence. We chose this variance experimentally so that the noise is visible, but the video is not useless. We choose N_w as in Table 6 to obtain $P_D = 0.999$ and $P_F = 0.001$. Table 11 shows $\hat{Q}$, σ_w, m_ψ, P_D and P_F after the additive white noise attack. The results shows that the proposed algorithm is robust to an additive white noise, $\mathcal{N}(0, 0.001)$, attack without increasing N_w.

Finally, we examine a trivial deliberate attack to erase the watermark. The adversary does not know which coefficients from the set CV have been watermarked, nor does he know the value of the watermark inserted in each watermarked coefficient. Recall that the set CV is the set of coefficients with visual watermarking capacity. To be confident that he has erased the watermark, he has to modify all the coefficients in CV. Since he does not know the value of the watermark, a trivial way to erase the

Table 11: Experimental results after the additive white noise attack when the target is to achieve $P_D = 0.999$ and $P_F = 0.001$ for a location-aware detector.

Sequence	$\hat{Q}$	σ_w	N_w	m_ψ	P_D	P_F
CARPHONE	14.39	59.17	645	6.19	0.9989	0.0009
CLAIRE	13.26	81.77	1395	6.08	0.9988	0.0014
MOBILE	15.33	51.86	435	6.18	0.9991	0.0014
MOTHER	12.84	63.65	928	6.14	0.9984	0.0015
SALESMAN	14.32	54.60	554	6.19	0.9994	0.0009
TABLE	14.84	54.45	604	6.14	0.9990	0.0016

watermark is to add the H.264 quantization step size, Q, to all the coefficients in CV or randomly add $+Q$ or $-Q$ to all the coefficients in CV. In both cases, he can only erase half of the watermark bits, and at the same time he increases the strength of remaining watermark bits by a factor of two. Since the detector response is a sum of the products of the watermark and watermarked coefficients scaled by their H.264 quantization step size, the average value of the detector response remains the same. The watermark detector can detect the watermark with the same performance without increasing N_w. Figure 17 shows a watermarked I-frame on the left and attacked version of that I-frame on right. The visual quality of the attacked I-frame is not severely degraded because the adversary only modified the coefficients with visual watermarking capacity. However, there are some small artifacts around the right side of the person's face, the left side of color of his coat and more artifacts are visible in the white and blue slats on the left. Note that in the watermarked frame 13.4% of coefficients in CV were watermarked with a strength of Q, and the attacked frame corresponds to a frame that has 86.6% of its coefficients in CV watermarked with a strength of Q and 6.7% of its coefficients watermarked with the strength of $2Q$. Therefore, we expect that the visual quality of the attacked I-frame to be worse than the watermarked I-frame. This agrees with our earlier observation that inserting the watermark in all the coefficients in CV results in visible artifacts.

We have compared our detection scheme with soft decision Viterbi decoding. Our results show that the detection performance is better when $N = 1500$ than when we construct a rate 1/3 convolutional code with $N = 500$.

(a) Watermarked I-frame (b) Attacked I-frame

Figure 17: Comparison of the visual quality after an adversary's attempt to remove the watermark by randomly adding $+Q$ or $-Q$ to all the coefficients in CV.

4.3 Discussion

We showed that the problem of watermark detection for video signals is different from images because of the large watermarking capacity in videos. By appropriately choosing the number of coefficients to compute the detector response, we can achieve any probability of detection and false alarm.

In our experiments, we assume that we know the values of $\hat{Q}$ and σ_w. If memory is not an issue, the owner can save these values in his records for different video sequences. Furthermore, if computational complexity is not an issue, the detector can calculate σ_w. Otherwise, since they vary only over a small range, the detector can assume a minimum value for $\hat{Q}$, and a maximum value for σ_w and choose N_w accordingly.

We assume that the detector can synchronize after any attack that causes desynchronization. However, if the original video sequence is not available, appropriate synchronization templates [3] are required to synchronize. Furthermore, we have not considered the effect of a self-collusion attack, which is one of the most powerful attacks for video. Our technique presented in Chapter 2 can be used as one solution to combat the self-collusion attack. The watermark can be embedded in different coefficients in CV by using the public and secret key proposed in Chapter 2. In [39], we showed that a similar watermark embedding technique to the technique in this paper is fairly robust to H.264 requantization. However, because of the computational complexity of this attack, it was not investigated in this chapter. Furthermore, we only showed the robustness of our algorithm to a trivial deliberate attack. Implementing more complex adversarial attempts to erase the watermark is a subject of further study.

4.4 Conclusion

In this chapter, we built a theoretical framework for watermark detection using a likelihood ratio test and used it to obtain video watermark detection with controllable detection performance. We detected the watermark by multiplying the possibly watermarked DCT coefficients of the decoded frame by the original watermark bits, calculating the sum of those terms, normalizing the result, and comparing it with a threshold. We have proved that the performance of the watermark detector only depends upon the conditional mean of the detector response under the hypothesis that the watermark exists in the DCT coefficients. This mean depends on three parameters: the H.264 quantization step size, the standard deviation of the watermarked DCT coefficients, and the number of watermarked DCT coefficients used to compute the detector response, N_w. We cannot control the first two parameters, however, we can control N_w. This is not the case with images, since there is a limited number of coefficients that can be watermarked in each image before the watermark is visible. Therefore, our video watermark detection algorithm calculates the number of DCT coefficients needed to compute the detector response to obtain the desired probability of detection, P_D for a given probability of a false alarm P_F. Our simulation results show that the theoretically chosen value for N does lead to the desired P_D and P_F in Monte Carlo trials. Furthermore, even after watermarking attacks, we can obtain any P_D and P_F by making worst case assumption on the first two parameters and computing N_w accordingly. The result of this work appears in [41, 42].

CHAPTER V

WATERMARK EMBEDDING WITH CONTROLLED VIDEO BIT RATE INCREASE

Most video watermarking algorithms embed the watermark in I-frames, which are essential for the video signal, but refrain from embedding in P- and B-frames, which are highly compressed by motion compensation. However, P-frames appear more frequently in the compressed video and their watermarking capacity should be exploited, despite the fact that embedding a watermark signal in P-frames can increase the video bit rate significantly.

In this chapter, we show that by limiting the watermark to nonzero quantized ac residuals in P-frames, the video bit rate increase can be held to reasonable values [37, 40]. Since the nonzero quantized ac residuals in P-frames correspond to non-flat areas that are in motion, temporal and texture masking are exploited at the same time. We also show that by embedding the watermark only in nonzero quantized ac residuals that have spatial masking capacity in I-frames, the number of watermarked coefficients in I-frames doubles with an increase in video bit rate that is no greater than previous compressed-domain video watermarking algorithms.

5.1 Problems Associated with Embedding the Watermark in P-frames

The algorithms proposed in [41, 57, 76] embed the watermark in I-frames because I-frames are crucial for the video signal. Note that our algorithm in [41] is the same as the algorithm presented in Chapter 3. Furthermore, P- and B-frames have less capacity because they are highly compressed by motion compensation. However, P-frames occur more frequently than I-frames in the compressed video. We used

Table 12: Percentage increase in video bit rate and the average number of watermarked coefficients per frame when I- or P-frames are watermarked using the algorithms in [41] and [76] for the scenario that the watermark is embedded in the encoder.

	Percentage increase in video bit rate				Average number of watermarked coefficients			
	I		P		I		P	
Sequence	[76]	[41]	[76]	[41]	[76]	[41]	[76]	[41]
CARPHONE	3.85%	5.60%	12.40%	18.27%	588	851	527	750
CLAIRE	7.90%	10.56%	17.35%	20.94%	270	342	252	303
MOBILE	5.47%	5.01%	19.32%	16.93%	2664	2252	2315	1944
MOTHER	4.99%	9.22%	8.92%	20.58%	261	499	229	447
SALESMAN	3.42%	5.04%	10.88%	17.69%	597	893	413	621
SOCCER	1.83%	2.54%	4.50%	7.73%	430	609	399	589
TABLE	3.44%	3.60%	12.30%	11.88%	894	788	557	529
TEMPETE	3.66%	4.20%	14.01%	18.23%	1328	1682	1215	1528
Average	4.32%	5.72%	12.46%	16.53%	879	990	738	839

the algorithms proposed in [41, 76] to embed the watermark in I- or P-frames. Our simulation results show that we can embed the watermark in a large number of coefficients in P-frames while preserving high visual quality, but the video bit rate can increase significantly. We set $T_{block} = 2$ and $T_{coef} = 10$ for the algorithm in [41] using a group of picture (GOP) structure of I B P B P B I, which has twice as many P-frames as I-frames. Since the video sequences are short, we used this GOP structure to obtain a sufficient number of watermarked I-frames, but a typical H.264 GOP would have more P-frames. Tables 12 and 13 show the increase in video bit rate and the average number of watermarked coefficients in an I- or P-frame when the watermark is embedded in the encoder and bitstream, respectively.

Tables 12 and 13 show that the increase in video bit rate is very significant when P-frames are watermarked for both watermark embedding scenarios. This increase in video bit rate has two sources. First, there is an increase in the bit rate of the macroblocks that were watermarked. Changing zero-valued coefficients to nonzero

Table 13: Percentage increase in video bit rate and the average number of watermarked coefficients per frame when I or P-frames are watermarked using the algorithms in [41] and [76] for the scenario that the watermark is embedded in the bitstream.

	Percentage increase in video bit rate				Average number of watermarked coefficients			
	I		P		I		P	
Sequence	[76]	[41]	[76]	[41]	[76]	[41]	[76]	[41]
CARPHONE	1.80%	3.19%	9.95%	15.10%	589	858	515	731
CLAIRE	4.50%	5.92%	14.97%	18.04%	265	339	251	297
MOBILE	1.96%	2.33%	16.96%	15.18%	2668	2259	2299	1934
MOTHER	2.95%	6.59%	7.78%	18.30%	268	533	232	446
SALESMAN	1.79%	2.94%	9.73%	15.83%	607	915	417	630
SOCCER	0.97%	1.48%	3.40%	5.85%	435	634	396	590
TABLE	1.26%	1.83%	9.98%	9.96%	895	790	552	533
TEMPETE	1.59%	2.09%	11.52%	15.32%	1331	1688	1209	1523
Average	2.10%	3.29%	10.53%	14.19%	882	1002	734	835

values can significantly increase the video bit rate because H.264 uses a run-length code. Since the number of nonzero coefficients in P-frames is very small compared to I-frames, there is a high probability that a zero coefficient changes to a nonzero coefficient. Thus, the increase in video bit rate is more significant when P-frames are watermarked. Second, if the watermark is embedded in the encoder, there will be an increase in the bit rate of the macroblocks that are predicted from the watermarked macroblocks. This is confirmed by comparing the results of Table 12 and 13, which shows that the increase in video bit rate when the watermark is embedded in the bitstream is smaller than when it is embedded in the encoder.

The reason for the slight difference in the number of watermarked coefficients in I-frames when the watermark is embedded in the encoder for the algorithms in [41, 76] with the values reported in Chapter 3 is as follows. H.264 has an option of using both 4×4 and 8×8 transforms. The watermark detection algorithm in Chapter 4 [41] computes the location of watermarked coefficients based on Watson's human visual model assuming that all blocks use a 4×4 transform, since the majority of the blocks usually use 4×4 transforms. However, the information about the transform used for each block is no longer available after the video is decoded. Therefore, the watermark detector assumes that the coefficients in 8×8 transformed blocks are watermarked and uses them with other watermarked coefficients to detect the watermark. This causes the value of $\hat{Q}$ used in the watermark detector to be further from the quantization step size (Please see Chapter 4 for explanation of $\hat{Q}$.) In this chapter, to have a fair comparison of the actual number of watermarked coefficients in these algorithms, and because our new watermark detection algorithm does not depend on the location of watermarked coefficients, we do not count the number of coefficients in 8×8 transformed blocks that the watermark detector thinks are embedded with a watermark.

5.2 Proposed Method

The significant increase in video bit rate when the watermark is embedded in P-frames is probably unacceptable for video watermarking applications; thus, it is important to develop a watermarking scheme for P-frames than can hold the increase in video bit rate to reasonable values. To control the increase in video bit rate, it is important to not change the zero-valued coefficients in P-frames. Therefore, our proposed watermarking algorithm only embeds the watermark in nonzero quantized ac residuals in P-frames. Since the number of nonzero quantized ac residuals that have spatial masking capacity in P-frames is small, we embed the watermark in all the nonzero quantized ac residuals without using Watson's human visual model. We do not embed the watermark in dc coefficients because the human eye is highly sensitive to the dc value. Furthermore, embedding the watermark in dc coefficients increases the computed video sequence variance significantly. To maintain the performance of our watermark detector presented in Chapter 6, which is a variation of the watermark detector in Chapter 4, the number of watermarked coefficients would have to be increased so much as to negate any benefit from using the dc coefficients.

In Chapter 3, when embedding the watermark in I-frames, we saw that the algorithms in [41, 57, 76] embed the watermark in a subset of coefficients in CV (coefficients with visual capacity for watermark embedding) because embedding in all of the coefficients in CV creates visible artifacts. In this chapter, when embedding the watermark in I-frames, we propose embedding the watermark in the coefficients in CV only if their corresponding quantized DCT residuals are non-zero. In the simulation results section, we show that this will double the number of watermarked coefficients in I-frames without an additional increase in video bit rate while preserving the perceptual quality of the video.

The watermark is a two-dimensional bipolar matrix, $W \in \{-1, 1\}$, the same size as the video frame with mean zero and variance one. However, the watermark embedder inserts the watermark in a subset of the DCT coefficients, CW, as described before. The watermark is inserted into those coefficients as

$$\tilde{R}'_{ijk} = \tilde{R}_{ijk} + W_{ijk}, \tag{38}$$

where $\tilde{R}_{ijk}$ is the quantized residual and $\tilde{R}'_{ijk}$ is the watermarked quantized residual. Thus,

$$R'_{ijk} = (\tilde{R}_{ijk} + W_{ijk}) \times Q_{ijk}, \tag{39}$$

and

$$I'_{ijk} = I_{ijk} + W_{ijk} Q_k, \tag{40}$$

where R'_{ijk} is the watermarked unquantized residual, Q_{ijk} is the quantization step size for coefficient ij in block k, I_{ijk} is the original DCT coefficient, and I'_{ijk} is the watermarked DCT coefficient. The addition of the watermark to the quantized DCT residuals is the same as the addition of the watermark times the quantization step size to the original DCT coefficients. When common signal processing operations or watermarking attacks on the video change the prediction mode of the block, the residual and prediction will change. However, if the video quality is still acceptable, the watermark can still be extracted from the decoded video sequence.

Embedding the watermark in all the nonzero quantized ac residuals without using Watson's human visual model in P-frames does not sacrifice visual quality because these coefficients correspond to non-flat moving areas. To show that the number of nonzero quantized ac residuals in each frame correspond to the amount of motion, we compute the global motion intensity in each P-frame. This is found by computing the weighted average magnitude of the motion vectors of all 4×4 blocks as follows:

$$MV_G(f) = \frac{1}{W_4 H_4} \sum_{w_4=1}^{W_4} \sum_{h_4=1}^{H_4} \frac{|mv_f(w_4, h_4)|}{|\text{ref}_f(w_4, h_4) - f|}, \tag{41}$$

where $|mv_f(w_4, h_4)|$ is the magnitude of the motion vector of block (w_4, h_4) at frame f, and $\text{ref}_f(w_4, h_4)$ is the reference frame number for this block. W_4 and H_4 are the width and height of the frame measured in 4×4 blocks. Figure 18 shows the global motion intensity of eight standard video sequences.

The plots show that the video sequences CLAIRE, MOTHER and SALESMAN are mostly static with limited motion in some frames. This is expected since these are videos of a person on a static background who occasionally moves his head or hand while talking. The video sequence CARPHONE shows a person in a moving vehicle. Therefore, the background moves frequently and the subject moves his whole upper body twice towards the end of the video. Thus, it is not surprising that the motion intensity plot of this video sequence shows higher motion. The video sequences MOBILE and TEMPETE have slow continuous motion, and the motion intensity plots demonstrate that. The video sequence SOCCER has considerable motion; thus the motion intensity of all of its frames is at least 250 and reaches as high as 2500. The video sequence TABLE shows the variation in motion intensity very well. This video sequence starts with movement of a table tennis player's hand followed by a scene

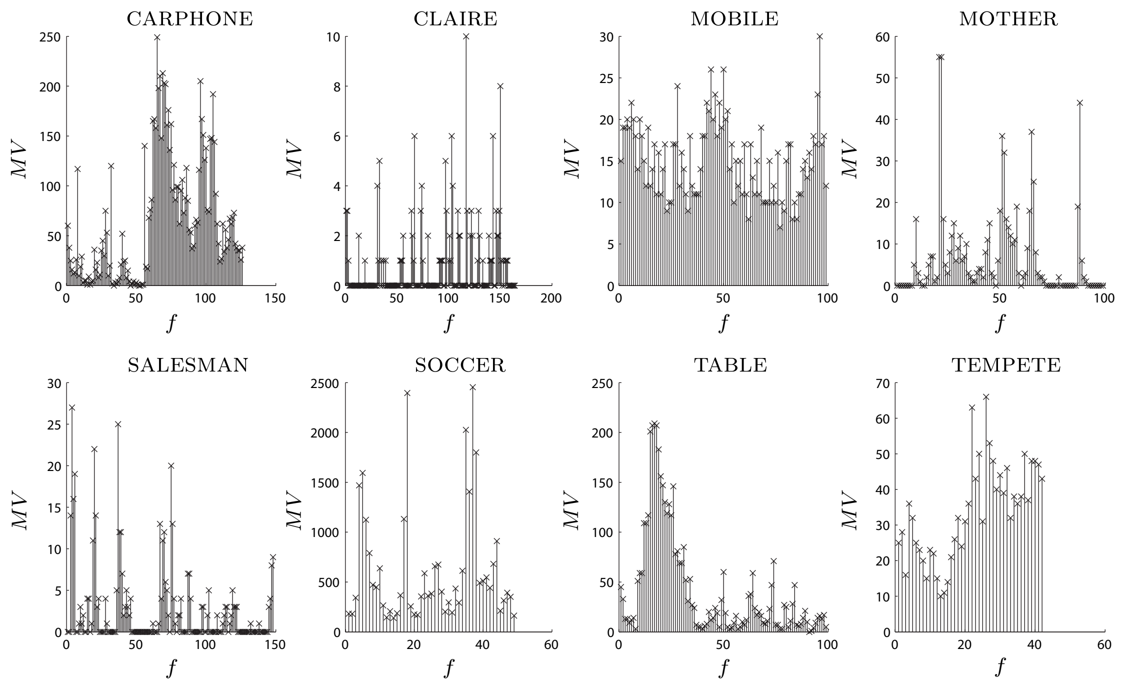


Figure 18: Weighted average magnitude of motion vectors, MV_G, in every P-frame of eight standard video sequences. Notice that the vertical scales vary significantly from image to image.

change. Figure 19 shows the number of nonzero quantized ac residuals, ν, which correspond to the number of watermarked coefficients in every P-frame of the eight standard video sequences. Comparing Figures 18 and 19 shows that the plots of the number of nonzero quantized ac residuals resemble the global motion intensity patterns.

5.3 Simulation Results

We implemented our proposed watermarking algorithm in the H.264 reference software version JM10.2 [1]. We used eight standard video sequences in QCIF format (176×144) at the rate of 30 frames per second for our simulation results. We compare the video bit rate increase and the perceptual quality using the visual quality metric (VQM) of the algorithm proposed in this Chapter with the algorithms in [41, 76].

We computed the video bit rate increase in our algorithm when only I-frames

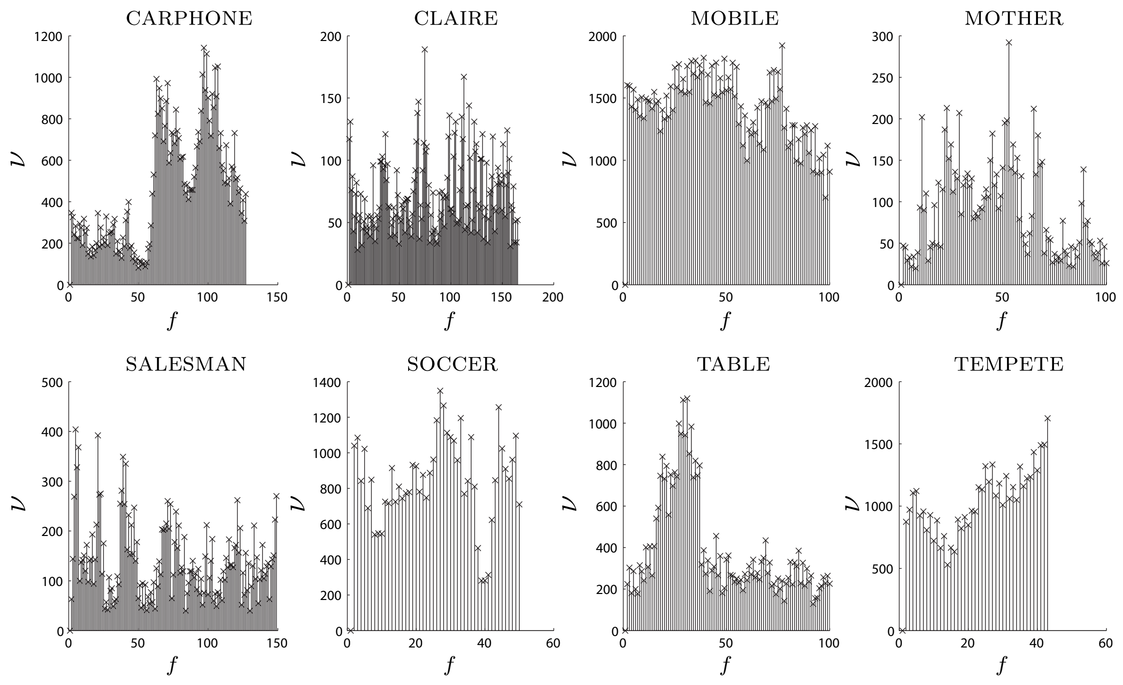


Figure 19: Number of nonzero quantized ac residuals, ν, in every P-frame of eight standard video sequences. Notice that the vertical scales vary significantly from image to image.

were watermarked and when only P-frames were watermarked. Table 14 shows the video bit rate increase and the average number of watermarked coefficients for our proposed algorithm when the watermark is embedded in the encoder and bitstream. Comparing this table with Table 12 shows that by embedding the watermark in only nonzero quantized ac residuals in P-frames instead of using the human visual model, the increase in video bit rate reduces from 12.46% in [76] and 16.53% in [41] to 1.54%, when the watermark is embedded in the encoder. This reduces the average number of watermarked coefficients in a P-frame from 738 in [76] and 839 in [41] to 559. Comparing this table with Table 13 shows that by embedding the watermark in only nonzero quantized ac residuals in P-frames, the increase in video bit rate reduces from 10.53% in [76] and 14.19% in [41] to 0.66%, when the watermark is embedded in the bitstream. This reduces the average number of watermarked coefficients in a P-frame from 734 in [76] and 835 in [41] to 542. Moreover, by embedding the watermark only in nonzero quantized ac residuals with spatial masking capacity in I-frames, we can

Table 14: Percentage increase in video bit rate and the average number of watermarked coefficients per frame when the watermark is embedded in the encoder and bitstream.

	Percentage increase in video bit rate				Average number of watermarked coefficients			
	Encoder		Bitstream		Encoder		Bitstream	
Sequence	I	P	I	P	I	P	I	P
CARPHONE	4.54%	1.56%	1.17%	0.27%	1235	461	1199	449
CLAIRE	7.40%	1.89%	1.81%	1.46%	474	73	460	69
MOBILE	6.28%	1.22%	2.54%	0.75%	6191	1426	6197	1388
MOTHER	5.93%	2.92%	2.04%	2.30%	501	92	476	87
SALESMAN	3.58%	1.55%	1.00%	1.15%	1106	140	1083	134
SOCCER	2.14%	0.83%	0.41%	−1.04%	813	841	799	822
TABLE	4.02%	0.98%	1.05%	0.23%	2038	390	2037	379
TEMPETE	4.27%	1.37%	1.64%	0.20%	2629	1052	2609	1010
Average	4.77%	1.54%	1.45%	0.66%	1709	559	1857	542

watermark twice as many coefficients in I-frames as [41, 76] with a similar increase in video bit rate for both watermark embedding scenarios.

Table 14 gives the percentage increase in video bit rate and the average number of watermarked coefficients in an I- or P-frame. However, since there are two parameters, it is hard to compare how different algorithms perform when embedding the watermark in I- or P-frames. We define the watermark cost, δ, as the increase in the number of bits used to encode the watermarked video per watermark bit

$$\delta = \frac{TB_{watermarked} - TB_{orig}}{\sum\limits_{f=1}^{L_f} N_w(f)}, \tag{42}$$

where $TB_{watermarked}$ is the number of bits used to code the watermarked video sequence, TB_{orig} is the number of bits used to code the original video sequence, and $\sum\limits_{f=1}^{L_f} N_w(f)$ is the total number of watermarked coefficients in that video sequence. In Tables 15 and 16, we have calculated watermark cost, δ, when the watermark is embedded in the encoder and bitstream, respectively.

Table 15: Watermark cost, δ, when the watermark is embedded in the encoder.

	I-frame			P-frame		
Sequence	[76]	[41]	New	[76]	[41]	New
CARPHONE	2.75	2.76	1.54	4.94	5.12	0.71
CLAIRE	4.53	4.77	2.41	5.26	5.27	1.98
MOBILE	2.15	2.33	1.06	4.32	4.67	0.44
MOTHER	4.01	3.87	2.48	4.04	4.78	3.40
SALESMAN	1.95	1.92	1.10	4.49	4.86	1.88
SOCCER	2.54	2.50	1.58	3.33	3.86	0.29
TABLE	1.60	1.90	0.82	4.54	4.60	0.52
TEMPETE	2.29	2.08	1.35	4.79	4.95	0.54
Average	2.72	2.76	1.54	4.46	4.76	1.22

Table 16: Watermark cost, δ, when the watermark is embedded in the bitstream.

	I-frame			P-frame		
Sequence	[76]	[41]	New	[76]	[41]	New
CARPHONE	1.28	1.56	0.41	4.06	4.34	0.13
CLAIRE	2.62	2.70	0.60	4.55	4.63	1.62
MOBILE	0.77	1.08	0.43	3.82	4.07	0.28
MOTHER	2.31	2.59	0.89	3.48	4.26	2.70
SALESMAN	1.00	1.10	0.31	3.98	4.28	1.48
SOCCER	1.34	1.40	0.31	2.51	2.90	-0.37
TABLE	0.58	0.96	0.21	3.71	3.84	0.12
TEMPETE	0.99	1.03	0.52	3.95	4.17	0.08
Average	1.36	1.55	0.46	3.75	4.06	0.75

These tables show several facts.

- The watermark cost, δ, of the algorithm in this chapter is significantly smaller than the algorithms in [41, 76] when embedding the watermark in both I- and P-frames for both scenarios.
- The watermark cost, δ, of the algorithms in [41, 76] is significantly smaller when embedding the watermark in I-frames than P-frames. Thus, the watermarking capacity of I-frames should be exploited completely before embedding the watermark in P-frames for these algorithms. However, we cannot conclude that embedding the watermark in either I- or P-frames has an advantage in terms of increase in video bit rate over the other for the algorithm proposed in this chapter.
- The watermark cost, δ, is significantly smaller when embedding the watermark in the bitstream than the encoder for all algorithms.

Table 15 shows the watermark cost, δ, of our proposed algorithm in this chapter versus the algorithms in [41, 76] when the watermark is embedded in both I- and P-frames. Our simulation results show that the watermark cost, δ, of our algorithm in this chapter is 1.50 versus 3.96 for [76] and 4.16 for [41] when the watermark is embedded in the encoder. The δ of our algorithm reduces to 0.51 when the watermark is embedded in the bitstream compared to 2.83 and 3.08 for the algorithms in [41, 76], respectively. The average increase in video bit rate of the proposed algorithm is 8.26% and 2.42% when the watermark is embedded in encoder and bitstream, respectively.

To compare the perceptual quality of our proposed watermark embedding algorithm in this chapter with the algorithms in [41, 76], we calculated the video quality metric (VQM) for these algorithms. The VQM software compares an original video clip and a processed video clip and reports a video quality metric (VQM) that correlates to perception. This metric is between zero and one with zero being no impairment and one being nominally maximum impairment. We used the compressed

Table 17: Watermark cost, δ, when the watermark is embedded in both I- and P-frames.

	Encoder			Bitstream		
Sequence	[76]	[41]	New	[76]	[41]	New
CARPHONE	4.21	4.39	1.34	3.03	3.23	0.35
CLAIRE	4.97	5.39	2.40	3.89	3.91	0.96
MOBILE	3.87	3.97	1.11	2.69	2.92	0.38
MOTHER	4.33	4.68	2.66	3.15	3.64	1.46
SALESMAN	3.54	3.66	1.56	2.76	2.97	0.65
SOCCER	3.12	3.46	0.66	1.94	2.32	−0.16
TABLE	3.60	3.59	1.09	2.37	2.68	0.18
TEMPETE	4.09	4.17	1.18	2.83	2.97	0.33
Average	3.96	4.16	1.50	2.83	3.08	0.51

but not watermarked video sequences as the original video clips and the watermarked video sequences as the processed video clips. Table 18 shows the VQM metric for these algorithms when only I-frames and when only P-frames are watermarked. This table shows that the algorithm in this chapter can embed twice as many watermark coefficients in each I-frame as the algorithms in [41, 76] with similar perceptual quality. Moreover, the algorithm in this chapter has the same perceptual quality as the algorithms in [41, 76] for P-frame only watermarking, although it does not use the spatial human visual model for embedding in P-frames. The VQM metric when the watermark is embedded both in I- and P-frames is similar to when only I-frames are watermarked.

To compare the perceptual quality of our proposed watermark embedding algorithm in this paper with the algorithms in [41, 76], we calculated the video quality metric (VQM) [4] for these algorithms. The VQM software compares an original video clip and a processed video clip and reports a video quality metric (VQM) that correlates to perception. This metric is between zero and one with zero being no impairment and one being nominally maximum impairment. We used the compressed

Table 18: Visual quality metrics (VQM) for the proposed algorithm and the algorithms in [41, 76] compared to compressed, but not watermarked videos.

	I-frames			P-frames			I and P-frames		
Sequence	[76]	[41]	New	[76]	[41]	New	[76]	[41]	New
CARPHONE	0.12	0.13	0.13	0.04	0.04	0.04	0.12	0.13	0.13
CLAIRE	0.10	0.10	0.10	0.03	0.03	0.03	0.10	0.10	0.10
MOBILE	0.05	0.05	0.06	0.04	0.03	0.03	0.06	0.05	0.06
MOTHER	0.15	0.15	0.14	0.04	0.05	0.04	0.15	0.15	0.15
SALESMAN	0.14	0.14	0.15	0.04	0.04	0.04	0.14	0.14	0.15
SOCCER	0.08	0.08	0.09	0.05	0.05	0.04	0.08	0.09	0.09
TABLE	0.15	0.16	0.16	0.05	0.05	0.05	0.16	0.16	0.16
TEMPETE	0.08	0.07	0.08	0.04	0.05	0.04	0.07	0.07	0.08
Average	0.11	0.11	0.11	0.04	0.04	0.04	0.11	0.11	0.11

but not watermarked video sequences as the original video clips and the watermarked video sequences as the processed video clips. Table 18 shows the VQM for these algorithms when only I-frames, only P-frames and both I and P-frames are watermarked. This table shows that the algorithm in this paper can embed twice as many watermark coefficients in each I-frame as the algorithms in [41, 76] with similar perceptual quality. Moreover, the algorithm in this paper has the same perceptual quality as the algorithms in [41, 76] for P-frame only watermarking, although it does not use the spatial human visual model for embedding in P-frames. The VQM when the watermark is embedded both in I- and P-frames is similar to when only I-frames are watermarked.

Furthermore, we computed the VQM between the original video and the compressed video, which we denote as CM, and the VQM between the original videos and the watermarked-compressed videos when only I-frames, only P-frames and both I- and P-frames are watermarked. Table 19 shows the VQM metric for these cases. This table shows that the average VQM between the original and compressed videos for these eight video sequences is 0.14. This table also shows that the average VQM

Table 19: Visual quality metrics (VQM) for the compressed video, proposed algorithm and the algorithms in [41, 76] compared to original videos.

		I-frames			P-frames			I and P-frames		
Sequence	CM	[76]	[41]	New	[76]	[41]	New	[76]	[41]	New
CARPHONE	0.16	0.16	0.17	0.16	0.16	0.16	0.16	0.16	0.16	0.16
CLAIRE	0.13	0.13	0.13	0.13	0.12	0.13	0.13	0.13	0.12	0.13
MOBILE	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
MOTHER	0.16	0.18	0.17	0.17	0.17	0.17	0.16	0.17	0.16	0.16
SALESMAN	0.17	0.16	0.16	0.16	0.16	0.16	0.17	0.16	0.15	0.15
SOCCER	0.12	0.13	0.12	0.13	0.12	0.13	0.12	0.12	0.13	0.13
TABLE	0.21	0.20	0.20	0.21	0.21	0.21	0.21	0.20	0.21	0.20
TEMPETE	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
Average	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.13	0.13

between the original and watermarked-compressed videos when only I-frames, only P-frames and both I- and P-frames are watermarked is also 0.14, This shows that the visual quality of the watermarked-compressed videos is the same as the compressed videos when compared to the original video sequences, i.e. watermarking plus compression did not increase the perceived distortion over compression alone.

5.4 Conclusion

In this chapter, we explored watermark embedding in P-frames. The challenge in embedding the watermark in P-frames is that the video bit rate increases significantly. Thus, we only embedded the watermark in nonzero quantized ac residuals in P-frames. Since these coefficients correspond to non-flat areas that are in motion, temporal and texture masking were exploited at the same time. We showed that the number of nonzero quantized ac residuals in each frame resembles the motion intensity plots in different video sequences. We also proposed embedding the watermark in nonzero quantized ac residuals with spatial masking capacity in I-frames. Our simulation results showed that the bit rate increase per watermark bit of our algorithm in this chapter is significantly smaller than the previous compressed-domain video watermarking algorithms when embedding the watermark in both I- and P- frames. The result of this work appears in [37, 40].

CHAPTER VI

LOCATION-UNAWARE DETECTION (LUD)

In some applications, the watermark detector does not know which coefficients are embedded with the watermark. For example, our watermark embedding algorithm in Chapter 5 embeds the watermark in the nonzero quantized ac residuals. The identity of those locations is lost when the video is completely decoded. Furthermore, a reencoding or a deliberate attack that inserts additional nonzero coefficients would cause desynchronization and consequent failure in watermark detection. For these applications it is important that the watermark detection algorithm not depend on the precise location of the watermark signal.

Thus, the video watermark detection algorithm in this chapter looks at all the ac coefficients to detect the watermark [37, 42]. Our algorithm calculates the number of frames, F, required for the watermark detector to obtain the desired probability of a detection, P_D, for a given false alarm probability, P_F. This is not the case for images, since there is a limited number of coefficients that can be watermarked in each image before the watermark becomes visible. Our simulation results show that the theoretically chosen value for F does lead to the desired values of P_D and P_F in Monte Carlo trials.

6.1 Theoretical Framework

Our new watermark detection algorithm, which is a modification of the watermark detection algorithm in Chapter 4, looks at all the ac coefficients to detect the watermark. Assume that the total number of coefficients used to detect the watermark is N. From these N coefficients, the watermark signal will be embedded in N_w of them and N_o will not be watermarked. Thus,

$$N = N_w + N_o. \tag{43}$$

The observations under the two hypotheses are as follows:

$$\begin{aligned} H_0 : y_\ell &= I_\ell \qquad\qquad\qquad \forall\, \ell \\ H_1 : y_\ell &= \begin{cases} I_\ell & \ell \notin CW \\ I_\ell + W_\ell Q_\ell & \ell \in CW. \end{cases} \end{aligned} \tag{44}$$

where CW is the set of watermarked coefficients, I_ℓ is the DCT coefficient of the video frame, Q_ℓ is the H.264 quantization step size selected by the video encoder for that coefficient and W_ℓ is chosen from a bipolar watermark sequence with mean zero and variance one. The index ℓ denotes the ℓ^{th} watermark bit or the ℓ^{th} DCT coefficient.

The ac coefficients outside of CW are irrelevant to the detection problem and could have been discarded if the location of watermark signal were known. This can also be verified by writing the likelihood ratio test for all the ac coefficients. However, since the watermark detector does not know the exact locations of the watermarked coefficients, it can compute the detector response over all ac coefficients instead of only the possibly watermarked coefficients. This does not create a problem because the mean of the detector response over the coefficients outside CW is close to zero. In Chapter 4, we saw that the performance of the detector depends on the distance between the means of the detector response under the two hypotheses. Thus, we have

$$Y = \sum_{\ell=1}^{N} Y_\ell W_\ell Q_\ell \underset{H_0}{\overset{H_1}{\gtrless}} \lambda. \tag{45}$$

Expanding the summation, the detector response under the two hypotheses is

$$\begin{aligned} H_0 : Y &= \sum_{\ell=1}^{N} I_\ell W_\ell Q_\ell \\ H_1 : Y &= \sum_{\ell=1}^{N_w} (I_\ell + W_\ell Q_\ell) W_\ell Q_\ell + \sum_{\ell=1}^{N_o} I_\ell W_\ell Q_\ell. \end{aligned} \tag{46}$$

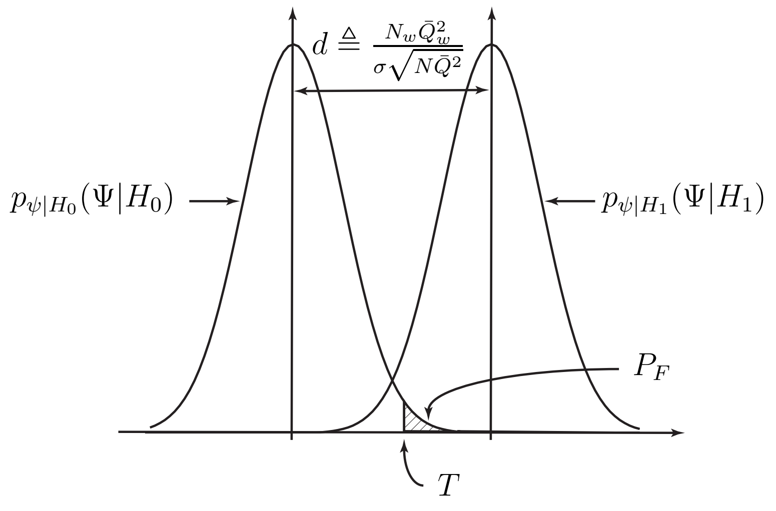
Therefore, Y is $\mathcal{N}(0, N\bar{Q}^2\sigma^2)$ under H_0, and Y is $\mathcal{N}(N_w\bar{Q}_w^2, N\bar{Q}^2\sigma^2)$ under H_1, where σ is the standard deviation of the ac coefficients in the video, $\bar{Q}^2 = (1/N)\sum_{\ell=1}^{N} Q_\ell^2$ and $\bar{Q}_w^2 = (1/N_w)\sum_{\ell=1}^{N_w} Q_\ell^2$. Note that although the mean of Y under the two hypotheses is equal to the mean of the detector response in (29), its variance is now $N\bar{Q}^2\sigma^2$ instead of $N_w\bar{Q}_w^2\sigma^2$. Multiplying (45) by $1/(\sigma\sqrt{N\bar{Q}^2})$ to normalize the Gaussian distribution gives

$$\psi = \frac{1}{\sigma\sqrt{N\bar{Q}^2}} \sum_{\ell=1}^{N} Y_\ell W_\ell Q_\ell \underset{H_0}{\overset{H_1}{\gtrless}} T, \tag{47}$$

where $T = \lambda/(\sigma\sqrt{N\bar{Q}^2})$. Then, ψ is $\mathcal{N}(0,1)$ under H_0, and ψ is $\mathcal{N}(\frac{N_w\bar{Q}_w^2}{\sigma\sqrt{N\bar{Q}^2}}, 1)$ under H_1. These probability densities are shown in Figure 20. The distance between the means of the two densities is now

$$d \triangleq \frac{N_w\bar{Q}_w^2}{\sigma\sqrt{N\bar{Q}^2}}. \tag{48}$$

Notice that if $Q_\ell = Q$ is fixed for all the DCT coefficients, then (48) simplifies to



$d \triangleq \frac{N_w\bar{Q}_w^2}{\sigma\sqrt{N\bar{Q}^2}}$
$p_{\psi|H_0}(\Psi|H_0)$
$p_{\psi|H_1}(\Psi|H_1)$
P_F
T

Figure 20: Probability densities $p_{\psi|H_0}(\Psi|H_0)$ and $p_{\psi|H_1}(\Psi|H_1)$ for a location-unaware detector.

$$d \triangleq \frac{N_w Q}{\sigma\sqrt{N}}. \tag{49}$$

If the ratio of watermarked coefficients to the total number of coefficients is equal to

ρ, then we can write (49) as

$$d \triangleq \frac{\rho\sqrt{N}Q}{\sigma}. \tag{50}$$

If we use the same embedding algorithm for the location-unaware detector as for the location-aware detector, the ratio of watermarked coefficients to the total number of coefficients, $\rho = \frac{N_w}{N}$ would be the same for both detectors. Then, comparing (50) with (35), we can conclude that to get the same detection performance (same d) for the same video sequence and same encoder parameters, we need to have

$$N_w = \frac{\grave{N}_w \sigma^2}{\rho \sigma_w^2}, \tag{51}$$

where N_w and $\grave{N}_w$ are the number of watermarked coefficients required to compute the detector response for the location-unaware detector and location-aware detector, respectively.

The above analysis assumed that the ratio of watermarked coefficients to the total number of coefficients, ρ, is a constant. However, if the watermark is embedded in nonzero quantized ac residuals, the number of watermarked coefficients may vary from frame to frame and consequently ρ will no longer be a constant. Consider N in (50) to be expressible as $N = F \times N_f$, where F is the number of frames required to compute the detector response and N_f is the number of ac coefficients in each frame for a certain video resolution. Assuming that $N_w(f)$ ac coefficients in frame f are watermarked, we can define the total number of watermarked coefficients in the ith group of F frames as

$$N_w^F(i) = \sum_{f=1}^{F} N_w(f + (i-1)F), \tag{52}$$

and

$$\rho^F(i) = \frac{N_w^F(i)}{FN_f} \qquad i = 1, 2, 3... \tag{53}$$

where $\rho^F(i)$ is the ratio of the number of watermarked coefficients to the total number of ac coefficients in F frames. Then, (50) becomes

$$d(i) \triangleq \frac{N_w^F(i)Q}{\sigma\sqrt{FN_f}}. \tag{54}$$

Thus, the probability distribution of the detector response under the hypothesis that the watermark exists, H_1, is no longer a normal Gaussian distribution, but is a Gaussian mixture model. We form the histogram of N_w^F divided into K clusters where the jth cluster, cl_j, has a centroid of N_j, and

$$p(N_w^F(i) \in cl_j) = \alpha_j \qquad j = 1, 2, ..., K, \tag{55}$$

where $\alpha_j > 0$ for $j = 1, 2, ...K$ and $\sum\limits_{j=1}^{K} \alpha_j = 1$. We have

$$\psi | N_w^F(i) \in cl_j \sim \mathcal{N}(\mu_j, 1), \tag{56}$$

where $\mu_j = \frac{N_j Q}{\sigma\sqrt{F N_f}}$. The distribution of ψ under hypothesis H_1 is a mixture of K Gaussian distributions

$$p(\psi | H_1) = \sum_{j=1}^{K} \alpha_j \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu_j)^2}{2}} \qquad j = 1, 2, ...K. \tag{57}$$

However, the probability distribution of ψ under the hypothesis that the watermark does not exist is still a normal Gaussian distribution $\mathcal{N}(0, 1)$. These probability distributions are shown in Figure 21. To evaluate the performance of the watermark detector, we need to compute the probability of detection, P_D, and the probability of false alarm, P_F. P_F is computed the same way as for the location-aware detector

$$P_F = \int_T^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \text{erfc}(T). \tag{58}$$

To compute P_D , we first define the probability of detection for each Gaussian distribution with mean μ_j as

$$P_{Dj} = \int_T^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu_j)^2}{2}} dx = \text{erfc}(T - \mu_j). \tag{59}$$

Then, P_D is

$$P_D = \sum_{j=1}^{K} \alpha_j \text{erfc}(T - \mu_j). \tag{60}$$

To achieve the specified value of P_D and P_F, the detector selects the threshold to agree with the value of P_F and then selects d to achieve the target values of P_D

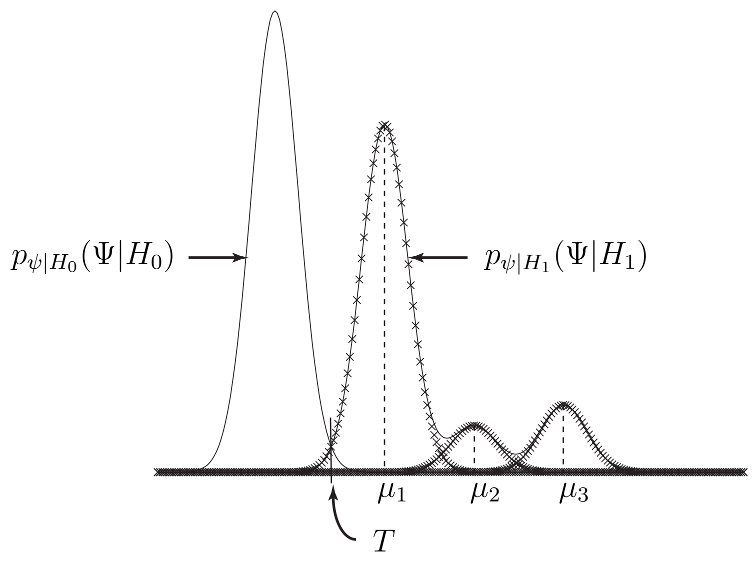


Figure 21: Illustrating the Gaussian mixture model and the probability densities $p_{\psi|H_0}(\Psi|H_0)$ and $p_{\psi|H_1}(\Psi|H_1)$ for the location-unaware detector.

assuming the probability distribution of the detector response under hypothesis H_1 is still a Gaussian distribution. We use the mean number of watermarked coefficients in each frame, $E\{N_w\}$ to roughly estimate the number of frames, F, required to get the desired performance from

$$d \triangleq \frac{E\{N_w\}\sqrt{F}Q}{\sigma\sqrt{N_f}}. \tag{61}$$

The computed value of F gives us a starting point. We calculate P_D from (60) and if P_D is smaller or greater than the desired P_D, we increase or decrease F by one until we reach the desired value of P_D. Our simulation results show that we reach the optimum value of F in a few iterations in most cases. Since knowing the optimum value of F does not compromise security or robustness of our watermarking algorithm, F can be made publicly available.

Our watermark detection scheme has several advantages. First, the error rate of the detector can be maintained regardless of the video sequence given that the video is long and the detector response latency can be arbitrary. However, the number of frames, F, required to obtain a certain detection performance depends on the histogram of the number of watermarked coefficients per frame in the video sequence. This means that the detector will produce results more frequently for some scenes or some videos than others. We believe that this is acceptable since nearly every video should have a sufficient number of watermarked DCT coefficients to produce detector responses at an acceptable rate. Another advantage is that if the watermark detector notices that the video sequence has been attacked to remove the watermark, it can increase the value of F to obtain more reliable detector responses. Notice that we are taking advantage of the large amount of data in video sequences compared to images to obtain more robust watermark detection. Another advantage is that computing the detector response, ψ, has low computational complexity. Recall that ψ is defined as

$$\psi = \frac{1}{\sigma\sqrt{N\bar{Q}^2}} \sum_{\ell=1}^{N} Y_\ell W_\ell Q_\ell. \tag{62}$$

The watermark sequence is a bipolar sequence of $\{-1, 1\}$, and usually Q_ℓ is constant within a subset of DCT coefficients that are used to compute one detector response. Thus, computing the detector response requires only the addition or subtraction of the DCT coefficients with few multiplications. Notice that the location-aware detector in Chapter 4 computed the sum in (62) over a smaller set of coefficients that are used to embed the watermark, N_w. However, finding the location of watermarked coefficients can have a high computational complexity, particularly if a human visual model is used. Thus, for some applications, where these locations have to be found every time a watermark is detected, the watermark detection algorithm in this section has a significantly lower complexity.

6.2 Simulation Results

To test the performance of our location-unaware detector (LUD), we used this detector to detect the watermark from P-frames of eight standard video sequences that were watermarked with the watermark embedding algorithm in Chapter 5. Subsection 6.2.1

gives the performance of the location-unaware detector and shows the robustness of the watermark embedding algorithm in Chapter 5 to several common signal processing attacks when LUD is used to detect the watermark. To compare the performance of location-aware and location-unaware detectors, we embedded the watermark in I-frames of six standard video sequences using our perceptual watermark embedding algorithm presented in Chapter 3. In Subsection 6.2.2, we compare the performance of our location-unaware detector with the location-aware detector presented in Chapter 4.

6.2.1 Watermark Detection from P-Frames

To evaluate the performance of location-unaware detector and to test the robustness of our watermark embedding algorithm in Chapter 5, we watermarked every P-frame of eight standard video sequences in the QCIF (176×144) format at the rate of 30 frames per second using the watermark embedding algorithm in Chapter 5. We implemented our watermark embedding algorithm in the H.264 reference software version JM10.2 [1]. The H.264 encoder used a fixed quantization step size $Q = 16$ and an intra period of 3 (group of pictures: I B P B P B I). We detected the watermark from P-frames of these video sequences using the location-unaware detector. To compare the experimental results with the theoretical framework for watermark detection derived in the previous section, a large number of watermarked coefficients are required to compute the detector response many times. Thus, we coded and watermarked every video sequence 100 times with an H.264 encoder. Note that the more detector responses we have, the more smoothly we can estimate their distribution.

In our first experiment, our goal is to obtain $P_F = 0.01$ and $P_D = 0.99$. From (58), we find that we can achieve $P_F = 0.01$ with $T = 2.325$. Then, to get a rough estimate of the number of frames required to achieve the desired P_D, we first assume P_D has a Gaussian distribution and we calculate d from (32) and we solve (61) to obtain a starting point for F. Then, we calculate the correct value of P_D from (60) and if P_D is not close to the desired P_D, we increase or decrease F until it is. Note that the difference between the watermarked coefficients from the decoded video with their values before the watermark was added is not exactly equal to the quantization step size because the encoding process is lossy. On average this difference is $\hat{Q}$ instead of Q, where $\hat{Q}$ is generally smaller, but close to, Q. Table 20 shows the standard deviation of the ac coefficients in the video sequence, σ, $\hat{Q}$, and the number of iterations, t, that it takes to find the number of watermarked frames, F, to calculate the detector response for the desired performance. It also shows the total number of detector responses $\#\psi$, the theoretically computed probability of detection from (60) that we denote as P_{Dt} , and the probabilities of detection, P_D, and false alarms, P_F found from Monte Carlo trials. Our results show that P_D is close to 0.99 and P_F is close to 0.01. Note that the larger $\#\psi$ is, the more exact P_D and P_F will be. In Figures 22 and 23, we plot the probability distribution of the detector response under H_0 and H_1 for the video sequences CARPHONE and MOBILE. We have a total of 839 detector responses for CARPHONE and 4949 detector responses for MOBILE from coding these video sequences 100 times. The symbols $\circ$ and $\diamond$ reflect the number of detector responses in the intervals centered around them. These figures show that the experimentally determined $p(\Psi|H_0)$ approximates a Gaussian distribution with a variance of one, however, $p(\Psi|H_1)$ no longer has a Gaussian distribution.

Now we show the effect of different attacks on detection performance. We first consider a 3×3 Gaussian filtering attack. We choose F and the threshold as in Table 20 to achieve $P_D = 0.99$ and $P_F = 0.01$. Table 21 gives σ, $\hat{Q}$, the theoretically determined P_{Dt} from (60), and P_D and P_F computed from the simulation results. Table 21 shows that after the Gaussian filtering attack both σ and $\hat{Q}$ become smaller. However, the reduction in the value of $\hat{Q}$ is more significant. Thus, for the same F as Table 20, $p(\Psi|H_1)$ moves towards $p(\Psi|H_0)$ because the mean value of ψ under

Table 20: Experimental results when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-unaware detector.

Sequence	$\hat{Q}$	σ	t	F	$\#\psi$	P_{Dt}	P_D	P_F
CARPHONE	15.18	15.99	13	15	839	0.9935	0.9940	0.0083
CLAIRE	14.88	15.68	7	104	159	0.9894	> 0.9937	0.0191
MOBILE	15.45	29.80	2	2	4949	0.9989	0.9988	0.0123
MOTHER	14.46	11.58	17	55	179	0.9895	0.9888	0.0056
SALESMAN	15.14	13.72	5	26	569	0.9901	0.9877	0.0141
SOCCER	15.02	13.62	3	3	1633	0.9960	> 0.9993	0.0104
TABLE	15.11	19.53	8	13	761	0.9926	0.9908	0.0158
TEMPETE	15.39	20.79	2	2	2099	0.9982	0.9962	0.0148

Table 21: Experimental results after the 3×3 Gaussian filtering attack when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-unaware detector.

Sequence	$\hat{Q}$	σ	F	P_{Dt}	P_D	P_F
CARPHONE	10.27	13.88	15	0.9844	0.9774	0.0083
CLAIRE	10.39	14.78	104	0.9620	0.9427	0.0191
MOBILE	9.88	24.91	2	0.9911	0.9808	0.0123
MOTHER	10.56	10.20	55	0.9819	0.9777	0.0056
SALESMAN	10.49	12.19	26	0.9816	0.9684	0.0141
SOCCER	10.51	12.43	3	0.9838	0.9890	0.0104
TABLE	10.23	15.82	13	0.9918	0.9803	0.0158
TEMPETE	10.54	17.84	2	0.9494	0.9228	0.0148

hypothesis H_1 becomes smaller. Since we set the threshold T as before, we still obtain the same P_F, but P_D is lower than its value when not under attack. Increasing F further, we can still achieve $P_D = 0.99$. Suppose that after the Gaussian filtering attack, $\hat{Q}$ becomes as small as 10. We use this value to calculate the required F for each video sequence. Table 22 shows the assumed value for $\hat{Q}$, the new values of F that give the desired $P_D = 0.99$, and P_D after the 3×3 Gaussian filtering attack.

We also look at a cropping attack. In our experiment, we crop each video frame to approximately 50% of its original size from the four sides. We assume that the detector can determine how the video is cropped by synchronization templates or by

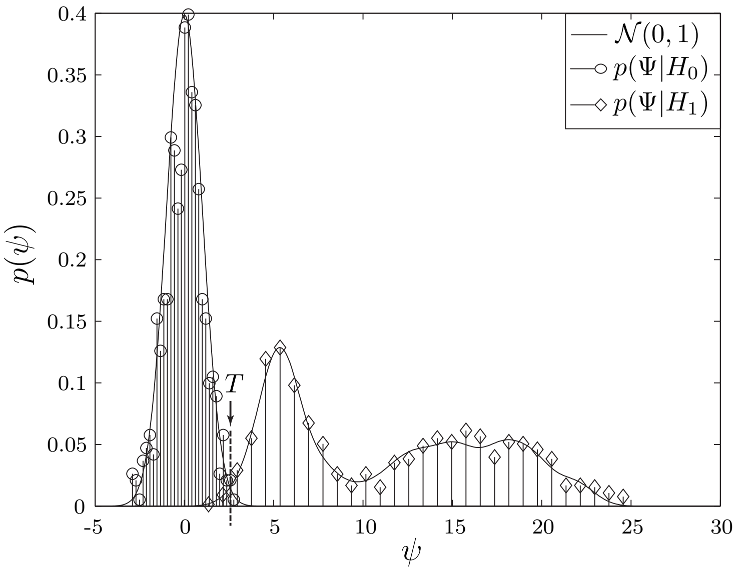


Figure 22: Detector response probability distribution of a location-unaware detector to achieve $P_D = 0.99$ and $P_F = 0.01$ for CARPHONE video sequence.

shifting the watermark array on top of each frame. If the detector can detect the watermark for one specific position of the watermark, then it can conclude that the watermark exists. The simulation results show that the detection performance even improves for some video sequences because cropping can increase ρ and therefore a smaller F suffices to give the desired P_D. This usually happens for the video sequences that have most of their activity in the middle of the frames like CLAIRE, MOTHER and SALESMAN. Since these video sequences have a small ρ (they have little motion and texture), they required a larger number of frames compared to the rest of the video sequences to detect the watermark under no attack. Thus, we do not want to increase F further for these video sequence after a cropping attack. Table 23 shows σ, $\hat{Q}$, the theoretically determined P_{Dt} from (60), and values of P_D and P_F from simulation results. In this case, we achieve the desired performance without increasing F, because most of the watermarked coefficients are in the middle of the video frames. If the watermark was uniformly distributed, ρ would remain the same after the cropping attack, and we would need twice as many frames as when under

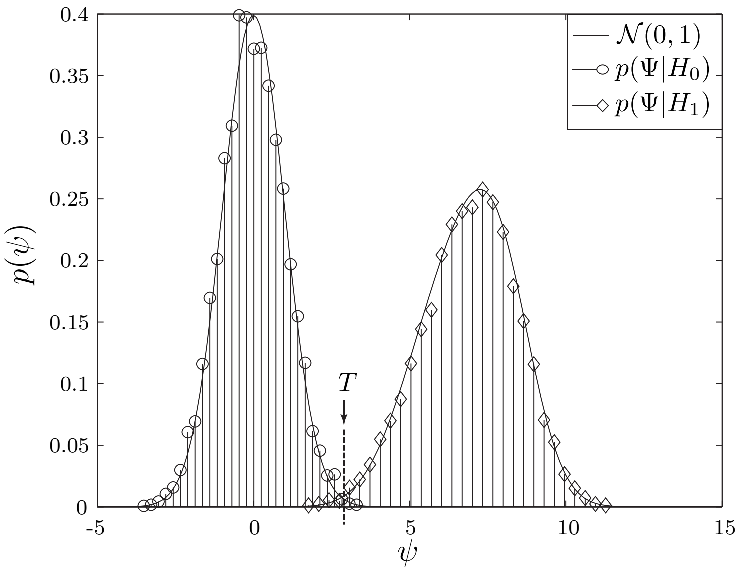


Figure 23: Detector response probability distribution of a location-unaware detector to achieve $P_D = 0.99$ and $P_F = 0.01$ for MOBILE video sequence.

no attack to obtain the same detection performance.

Next, we consider the effect of additive white noise. We add white noise of mean zero and variance 0.001 to each frame of the video sequence. We chose the variance experimentally so that the noise is visible, but the video is not useless. Table 24 shows σ, $\hat{Q}$, the theoretically determined P_{Dt} from (60), and values of P_D and P_F after the additive white noise attack. The simulations results show that the detector achieves the desired performance after the additive white noise attack without increasing F.

Finally, we examine the robustness of the proposed algorithm to a requantization attack. We change the quantization step size from 16 to 26. Table 25 shows σ, $\hat{Q}$, the theoretically determined P_{Dt} from (60), and values of P_D and P_F after the requantization attack. Studying Table 25 shows that $\hat{Q}$ becomes significantly smaller after the requantization attack, and this reduction is more significant for those video sequences that have fewer nonzero quantized ac residuals (less texture and motion). Although we do not obtain the desired probability of detection $P_D = 0.99$, the probability of detection is $P_D = 0.86$ on average for these video sequences after changing $Q = 16$

Table 22: Experimental results after the 3×3 Gaussian filtering attack with new values of F when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-unaware detector.

Sequence	$\hat{Q}$	σ	F	P_{Dt}	P_D	P_F
CARPHONE	10	13.88	17	0.9929	0.9906	0.0135
CLAIRE	10	14.78	137	0.9903	0.9832	0.0081
MOBILE	10	24.91	2	0.9919	0.9808	0.0123
MOTHER	10	10.20	61	0.9915	0.9938	0.0060
SALESMAN	10	12.19	30	0.9907	0.9899	0.0176
SOCCER	10	12.43	4	0.9949	0.9975	0.0098
TABLE	10	15.82	13	0.9912	0.9803	0.0158
TEMPETE	10	17.84	4	0.9957	0.9914	0.0148

Table 23: Experimental results after the cropping attack when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-unaware detector.

Sequence	σ	$\hat{Q}$	F	P_{Dt}	P_D	P_F
CARPHONE	15.22	16.65	15	0.9809	0.9797	0.0083
CLAIRE	15.01	13.79	104	0.9999	> 0.9937	0.0191
MOBILE	15.41	30.57	2	0.9652	0.9590	0.0123
MOTHER	14.50	9.40	55	0.9992	> 0.9944	0.0056
SALESMAN	15.17	15.56	26	0.9980	> 0.9982	0.0141
SOCCER	15.01	14.63	3	0.9371	0.9345	0.0104
TABLE	15.12	19.05	13	0.9981	0.9974	0.0158
TEMPETE	15.26	21.80	2	0.8726	0.8642	0.0148

Table 24: Experimental results after the additive white noise attack when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-unaware detector.

Sequence	$\hat{Q}$	σ	F	P_{Dt}	P_D	P_F
CARPHONE	15.17	17.79	15	0.9871	0.9845	0.0083
CLAIRE	14.76	17.43	104	0.9680	0.9618	0.0191
MOBILE	15.39	30.78	2	0.9984	0.9986	0.0123
MOTHER	14.45	13.96	55	0.9583	0.9497	0.0056
SALESMAN	15.44	15.79	26	0.9664	0.9701	0.0141
SOCCER	15.01	15.69	3	0.9912	0.9945	0.0104
TABLE	15.05	20.98	13	0.9848	0.9803	0.0158
TEMPETE	15.37	22.20	2	0.9964	0.9948	0.0148

Table 25: Experimental results after changing the quantization step size from 16 to 26 when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-unaware detector.

Sequence	$\hat{Q}$	σ	F	P_{Dt}	P_D	P_F
CARPHONE	7.58	15.29	15	0.9625	0.8868	0.0083
CLAIRE	5.28	15.20	104	0.7174	0.7389	0.0191
MOBILE	8.76	28.20	2	0.9900	0.9709	0.0123
MOTHER	3.46	10.84	55	0.7229	0.7039	0.0056
SALESMAN	5.63	12.70	26	0.8405	0.7944	0.0141
SOCCER	10.31	12.80	3	0.9922	0.9902	0.0104
TABLE	8.89	18.68	13	0.9543	0.9054	0.0158
TEMPETE	8.80	19.47	2	0.9875	0.9381	0.0148

Table 26: Experimental results after changing the quantization step size from 16 to 26 with new values of F when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-unaware detector.

Sequence	$\hat{Q}$	σ	F	P_{Dt}	P_D	P_F
CARPHONE	7.58	15.29	37	0.9999	0.9941	0.0083
CLAIRE	5.28	15.20	538	0.9999	> 0.9666	0.0191
MOBILE	8.76	28.20	3	0.9987	0.9945	0.0123
MOTHER	3.46	10.84	252	0.9999	> 0.9743	0.0056
SALESMAN	5.63	12.70	90	0.9999	0.9939	0.0141
SOCCER	10.31	12.80	3	0.9922	0.9902	0.0104
TABLE	8.89	18.68	37	0.9995	0.9945	0.0158
TEMPETE	8.80	19.47	4	0.9998	0.9933	0.0148

to $Q = 26$. Increasing F further, we can still achieve $P_D = 0.99$. Table 26 shows the new values of F that give the desired $P_D = 0.99$ after changing the quantization step size from 16 to 26.

6.2.2 Comparison of Location-Aware and Location-Unaware Detectors

We embedded the watermark in the H.264 reference software version JM10.2 [1] using our perceptual watermark embedding algorithm in Chapter 3. We used six standard QCIF video sequences (176×144) at the rate of 30 frames per second for our simulation. To compare the experimental results with the theoretical framework, a large number of detector responses is required. Thus, we coded and watermarked I-frames of every video sequence 80 times by an H.264 encoder with an intra period of one (group of picture: I B I). Note that the more detector responses we have, the more smoothly we can estimate their distribution. The H.264 encoder used a fixed quantization step size $Q = 16$ for I-frames. On average the difference between the watermarked coefficients and their values before watermarking is $\hat{Q}$ instead of Q because the encoding process is lossy. In this subsection, we detect the watermark from I-frames of those video sequences using location-aware detector (LAD) and location-unaware detector (LUD), and we compare the performance of those detectors. Note that the results for LAD case has been extensively reported in Chapter 4. However, we summarize those results here for comparison purposes.

The goal of our first experiment is to obtain $P_D = 0.99$ and $P_F = 0.01$. We first consider the LAD case. Solving equations (32) and (33) analytically, these probabilities can be achieved for $T = 2.325$ and $d = 4.65$. We detect the watermark by computing the detector response over N_w watermarked coefficients for each video. We use the values of $\hat{Q}$ and σ_w given in Table 27 to compute N_w from (35). We calculate P_D and P_F based on this threshold. The total number of detector responses obtained from coding each video sequence 80 times, $\#\psi$, the mean value of the detector response, m_ψ, P_D and P_F obtained from our experiments are shown in Table 27. This table shows that the theoretically chosen value for N_w does lead to the desired $P_D = 0.99$ and $P_F = 0.01$.

To obtain the same detection performance for the LUD case, we set the threshold as before; $T = 2.325$. To roughly estimate the number of frames required to achieve the desired P_D, we first assume P_D has a normal Gaussian distribution, we calculate d from (32) and we solve (61) to obtain a starting point for F. Then, we calculate the correct value of P_D from (60), and if P_D is not close to the desired P_D, we increase

Table 27: Experimental results when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-aware detector.

Sequence	$\hat{Q}$	σ_w	N_w	$\#\psi$	m_ψ	P_D	P_F
CARPHONE	14.40	59.17	365	31978	4.68	0.9898	0.0108
CLAIRE	13.52	81.73	790	10328	4.67	0.9905	0.0114
MOBILE	15.37	51.86	246	97432	4.71	0.9917	0.0121
MOTHER	12.91	63.65	526	12086	4.64	0.9877	0.0140
SALESMAN	14.33	54.60	314	44168	4.67	0.9913	0.0109
TABLE	14.95	59.45	342	24912	4.70	0.9920	0.0107

Table 28: Experimental results when the target is to achieve $P_D = 0.99$ and $P_F = 0.01$ for a location-unaware detector.

Sequence	$\hat{Q}$	σ	F	$\#\psi$	P_{Dt}	P_D	P_F
CARPHONE	14.40	16.06	1	14799	0.9859	0.9817	0.0101
CLAIRE	13.52	15.78	3	6399	0.9819	0.9706	0.0123
MOBILE	15.37	29.85	1	11599	0.9999	0.9999	0.0127
MOTHER	12.91	11.76	1	11599	0.9895	0.9888	0.0110
SALESMAN	14.33	13.88	1	17599	0.9993	0.9985	0.0122
TABLE	14.95	19.47	3	3866	0.9804	0.9889	0.0142

or decrease F until it is. Table 28 shows the standard deviation of the ac coefficients, σ, the number of frames, F, required to obtain the desired performance, and the theoretically computed probability of detection from (60) that we denote as P_{Dt}, and the probability of detection, P_D, and the probability of false alarms, P_F, obtained from Monte Carlo trials. Our results show that P_D is close to 0.99 and P_F is close to 0.01.

In the next experiment, the target is to obtain $P_D = 0.999$ and $P_F = 0.001$. From (32) and (33), these probabilities can be achieved with $T = 3.09$ and $d = 6.18$. Tables 29 and 30 show the average value of $\hat{Q}$, σ_w, N_w, σ, F for the LAD and LUD cases. Again our results show that P_D is close to 0.999 and P_F is close to 0.001 for both detectors. However, comparing the total number of detector responses, $\#\psi$, in Table 27 with Table 28 and Table 29 with Table 30 shows more frequent detector responses

Table 29: Experimental results when the target is to achieve $P_D = 0.999$ and $P_F = 0.001$ for a location-aware detector.

Sequence	$\hat{Q}$	σ	N_w	$\#\psi$	m_ψ	P_D	P_F
CARPHONE	14.40	59.17	645	18095	6.21	0.9992	0.0009
CLAIRE	13.52	81.73	1395	5848	6.20	0.9995	0.0014
MOBILE	15.37	51.86	435	55099	6.25	0.9990	0.0014
MOTHER	12.91	63.65	928	6856	6.16	0.9991	0.0015
SALESMAN	14.33	54.60	554	25033	6.20	0.9992	0.0009
TABLE	14.95	59.45	604	14105	6.23	0.9993	0.0016

Table 30: Experimental results when the target is to achieve $P_D = 0.999$ and $P_F = 0.001$ for a location-unaware detector.

Sequence	$\hat{Q}$	σ	F	$\#\psi$	P_{Dt}	P_D	P_F
CARPHONE	14.40	16.06	2	7399	0.9990	0.9993	0.0009
CLAIRE	13.52	15.78	7	2742	0.9998	0.9989	0.0015
MOBILE	15.37	29.85	1	11599	0.9999	0.9999	0.0009
MOTHER	12.91	11.76	3	3866	0.9999	> 0.9997	0.0010
SALESMAN	14.33	13.88	2	8799	0.9999	> 0.9998	0.0014
TABLE	14.95	19.47	7	1657	0.9982	0.9988	0.0018

can be obtained when the location of watermark is known.

In Figures 24 and 25, we plot the probability distribution of the detector response under H_0 and H_1 for the video sequence CARPHONE for both detectors. Figure 24 shows that for a location-aware detector, the experimentally determined $p(\Psi|H_0)$ and $p(\Psi|H_1)$ approximate a normal Gaussian distribution with a variance of one. This justifies our assumption that the detector response has a normal Gaussian distribution. Figure 25 shows that for a location-unaware detector, $p(\Psi|H_0)$ still has a normal Gaussian distribution, however, $p(\Psi|H_1)$ no longer has a normal Gaussian distribution. The greater the change in the number of watermarked coefficients from frame to frame, the further the distribution will be from normal Gaussian.

Finally, we look at the effect of different attacks on detection performance. We first consider a 3×3 Gaussian filtering attack. We choose N_w as in Table 29, and

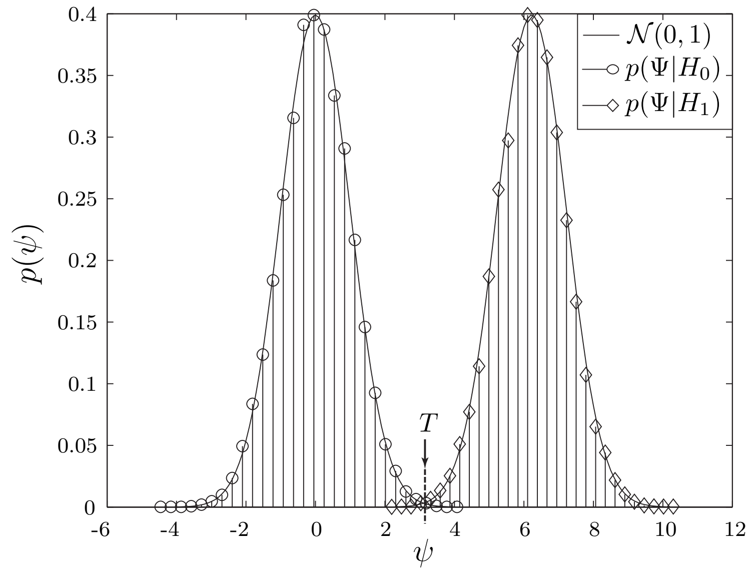


Figure 24: Detector response probability distribution of a location-aware detector to achieve $P_D = 0.999$ and $P_F = 0.001$ for CARPHONE video sequence.

F as in Table 30 and we set the threshold $T = 3.09$ to obtain $P_D = 0.999$ and $P_F = 0.001$ without an attack. Table 31 gives $\hat{Q}$, σ_w, σ and P_D obtained from both detectors after the Gaussian filtering attack. Comparing Table 31 with Tables 29 and 30 shows that after the Gaussian filtering attack, σ_w and σ remain approximately the same, but $\hat{Q}$ becomes significantly smaller. Thus, if we choose N_w and F as before, $p(\Psi|H_1)$ moves towards $p(\Psi|H_0)$ because the mean value of ψ under hypothesis H_1, m_ψ, becomes smaller. Since we set T as before, we still obtain the desired $P_F = 0.001$, but P_D is lower than 0.999.

Increasing N_w and F further, we can still achieve $P_D = 0.999$. Suppose that after the Gaussian filtering attack, $\hat{Q}$ becomes as small as nine, and suppose that the variance of the video sequences remains the same. We use these values to calculate the required N_w and F for each video sequence. Table 32 shows the calculated value for N_w and P_D for a location-aware detector. Since we assumed that $\hat{Q}$ gets smaller than it actually does and we set the threshold as before, we always have $P_D > 0.999$. We did not have any missed detections for any video sequence in our

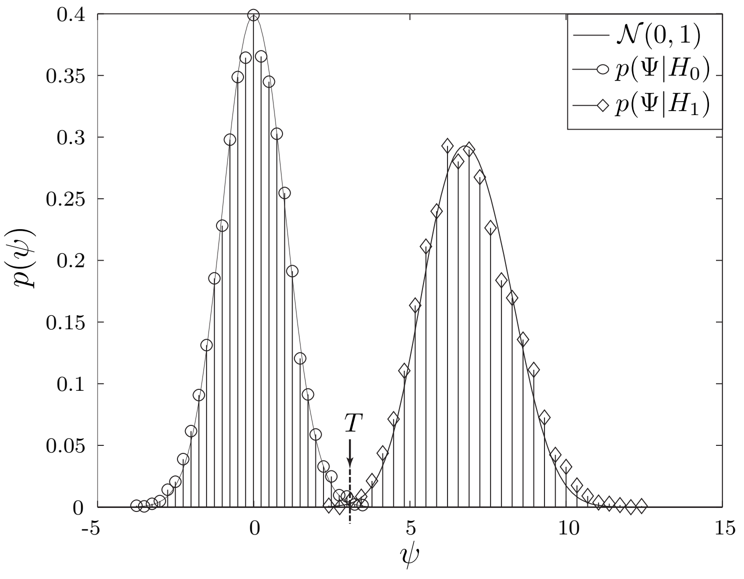


Figure 25: Detector response probability distribution of a location-unaware detector to achieve $P_D = 0.999$ and $P_F = 0.001$ for CARPHONE video sequence.

experiment. Therefore, the only statement that we can make is that the probability of missed detection, P_M, is smaller than one over the number of detector responses we computed, and P_D is greater than 1 minus this value. Table 32 also shows the number of frames F required to still achieve $P_D = 0.999$ and the P_D obtained from simulation results for the location-unaware detector.

We also apply a cropping attack by cropping each video frame to approximately 50% of its original size from the four sides. We assume that the detector can determine how the video is cropped by using either the original video sequence or synchronization templates. The simulation results show that the cropping attack does not affect the detection performance, however, it does affect the number of detector responses that can be extracted for each video sequence. For a location-aware detector, N_w is the same as before, however more frames are required to obtain N_w watermarked coefficients. For the location-unaware detector, we need twice as many frames to achieve the same performance as without an attack assuming that the watermarked coefficients are uniformly distributed within a frame. Note that the perceptual watermark

Table 31: Comparison of location-aware and location-unaware detectors after the 3×3 Gaussian filtering attack when the target is to achieve $P_D = 0.999$ and $P_F = 0.001$.

		LAD			LUD		
Sequence	$\hat{Q}$	σ_w	N_w	P_D	σ	F	P_D
CARPHONE	10.02	60.12	645	0.9118	13.90	2	0.9751
CLAIRE	9.55	81.01	1395	0.9300	14.83	7	0.9847
MOBILE	9.43	51.92	435	0.8300	24.84	1	0.9919
MOTHER	9.62	65.11	928	0.9400	10.28	3	0.9961
SALESMAN	10.08	54.57	554	0.9380	12.27	2	0.9998
TABLE	9.38	59.59	604	0.8500	15.73	7	0.9958

Table 32: Comparison of location-aware and location-unaware detectors after the 3×3 Gaussian filtering attack with new values of N_w and F when the target is to achieve $P_D = 0.999$ and $P_F = 0.001$.

	LAD			LUD		
Sequence	N_w	$\#\psi$	P_D	F	$\#\psi$	P_D
CARPHONE	1650	6220	> 0.9998	3	4933	0.9980
CLAIRE	3149	2511	> 0.9992	11	1745	> 0.9994
MOBILE	1268	17491	> 0.9999	2	5799	> 0.9998
MOTHER	1910	2848	> 0.9997	4	2899	0.9996
SALESMAN	1405	9546	> 0.9999	2	8799	0.9998
TABLE	1666	4734	> 0.9998	9	1288	> 0.9992

Table 33: Comparison of location-aware and location-unaware detectors after the cropping attack when target is to achieve $P_D = 0.999$ and $P_F = 0.001$.

		LAD			LUD		
Sequence	$\hat{Q}$	σ_w	N_w	P_D	σ	F	P_D
CARPHONE	14.53	60.54	645	0.9990	16.73	4	0.9962
CLAIRE	14.45	80.01	1395	0.9997	13.86	14	> 0.9992
MOBILE	15.42	49.81	435	0.9997	30.62	2	0.9991
MOTHER	12.77	50.06	928	0.9997	9.51	6	0.9444
SALESMAN	14.87	54.51	554	0.9996	15.77	4	> 0.9994
TABLE	14.92	59.99	604	0.9984	18.96	14	0.9964

embedding algorithm presented in Chapter 3 distribute the watermarked coefficients more uniformly than the watermark embedding algorithm presented in Chapter 5, which embeds the watermark in the nonzero quantized ac residuals. Table 33 shows N_w, F and P_D for both detectors. The total number of detector responses in this table is approximately half of their corresponding values in Table 29 and 30 since the video sequences have been cropped to approximately 50% of their original size. Note that we used the QCIF format (176×144) for our simulation results, which has one of the smallest resolutions; the results improve for higher resolution videos.

Finally, we consider the effect of additive white noise. We add white noise of mean zero and variance 0.001 to each frame of the video sequence. We chose the variance experimentally so that the noise is visible, but the video is not useless. We choose N as in Table 29 and F as in Table 30 to obtain $P_D = 0.999$ and $P_F = 0.001$. Table 34 shows $\hat{Q}$, σ_w, σ, and P_D of both detectors after the additive white noise attack. Our simulation results show that the proposed algorithm is robust to the additive white noise, $\mathcal{N}(0, 0.001)$, attack without increasing N_w and F.

6.3 Conclusion

For some watermark embedding algorithms like when embedding the watermark in nonzero ac residuals, the location of watermark is lost after the video is decoded.

Table 34: Comparison of location-aware and location-unaware detectors after the additive white noise attack when the target is to achieve $P_D = 0.999$ and $P_F = 0.001$.

		LAD			LUD		
Sequence	$\hat{Q}$	σ_w	N_w	P_D	σ	F	P_D
CARPHONE	14.39	59.17	645	0.9989	17.85	2	0.9922
CLAIRE	13.26	81.77	1395	0.9988	17.51	7	0.9960
MOBILE	15.33	51.86	435	0.9991	30.83	1	0.9999
MOTHER	12.84	63.65	928	0.9984	14.09	3	0.9979
SALESMAN	14.32	54.60	554	0.9994	15.93	2	0.9997
TABLE	14.84	54.45	604	0.9990	20.92	7	0.9982

Thus, it is important that the watermark detection algorithm not depend on the exact location of the watermark signal. In this chapter, we built a theoretical framework for a watermark detection algorithm based on a likelihood ratio test that does not depend on the exact location of watermarked coefficients. This framework was used to obtain video watermark detection with controllable detection performance. We used this detector to detect the watermark from P-frames of eight standard video sequences that were embedded with the watermark using our watermark embedding algorithm presented in Chapter 5. Our simulation results showed that the theoretically chosen value for F did lead to the desired values of P_D and P_F in Monte Carlo trials. We tested the robustness of our proposed algorithm to several common signal processing attacks such as filtering, 50% cropping, addition of white noise, $\mathcal{N}(0, 0.001)$, and a requantization attack. Our simulation results showed that our proposed algorithm was robust against these attacks. We also compared the performance of location-unaware detector with the location-aware detector presented in Chapter 4. While both detectors could achieve any probability of detection and false alarm, the location-aware detector could generate detector responses more frequently than the location-unaware detector. The result of this work appears in [37, 42].

CHAPTER VII

CONCLUSION

The goal of this dissertation was to present a robust watermarking algorithm for H.264 and to address challenges in compressed-domain video watermarking. In Section 7.1, we summarize our contributions and in Section 7.2, we suggest avenues for future study.

7.1 Contributions

Watermarking digital video introduces challenges that are not present when watermarking digital images. The large amount of data and inherent redundancy between frames makes video watermarking algorithms susceptible to self-collusion attacks. The self-collusion attack is one of the most powerful attacks for video. In Chapter 2, we designed a novel low complexity watermarking algorithm that was robust to self-collusion attacks [38]. The algorithm embedded the watermark in the quantized ac residuals of the H.264-compressed video. It achieved collusion resistance by embedding the watermark in the same location in similar frames and different locations in dissimilar frames. The coefficient within a macroblock that holds the watermark was determined by a key that was specific to that macroblock, but this could require a long key stream sequence. To avoid this problem, the key was generated using a public key extracted from features of the macroblock and the copyright owner's secret key. It was proposed that the relative difference of the DC coefficients of the 4×4 blocks in a macroblock is a robust feature for public key extraction.

The algorithm we proposed in Chapter 2 embedded the watermark in the compressed video, but this algorithm was not robust against several common watermarking attacks besides the self-collusion. The watermark was embedded in and extracted from the I-frame quantized ac residuals, so any simple processing, such as filtering followed by reencoding by an H.264 encoder, changes the intra-macroblock prediction modes, and thus the residuals, which makes watermark recovery impossible. In Chapter 3, we presented a perceptual watermarking algorithm for H.264 that was robust to common signal processing attacks [39, 41]. To achieve this goal we embedded the watermark in the residuals to avoid decompressing the video and also to reduce the complexity of the watermarking algorithm. However, the watermark was extracted from the decoded video sequence to make the algorithm robust to intra-prediction mode changes. Since H.264's high compression performance leaves little room for an imperceptible signal to be inserted, we employed a human visual model to increase the payload and add robustness while limiting visual distortion. Watson *et al.* derived a model for distortion perception in 8×8 DCT blocks [46, 72], and we extended this human visual model for the 4×4 DCT block used in H.264. If all the coefficients with visual capacity for watermark embedding were used, the visual quality of the video would be degraded. We proposed embedding the watermark in a selected subset of the coefficients that have visual watermarking capacity by using a key-dependent algorithm. This makes the algorithm more robust to malicious attacks. Furthermore, we designed our algorithm so that the watermark is spread over frequencies and blocks to reduce the error pooling effect described by Watson [72].

In Chapter 4, we presented our video watermark detection algorithm when the precise location of watermark signal is known to the detector, a detector we call location aware [41, 42]. We built a theoretical framework for watermark detection based on a likelihood ratio test and used it to obtain video watermark detection with controllable detection performance. We detected the watermark by multiplying the possibly watermarked DCT coefficients of the decoded frame by the original watermark bits, calculating the sum of those terms, normalizing the result, and comparing the sum against a threshold. We proved that the performance of our watermark detector only depends upon the conditional mean of the detector response under the hypothesis that the watermark exists in the DCT coefficients. This mean depends upon three parameters: the average of the squares of the H.264 quantization step sizes of watermarked DCT coefficients, the standard deviation of watermarked DCT coefficients, and the number of watermarked DCT coefficients, N_w, over which the detector response is computed. We cannot control the first two parameters, but we can control N_w. This is not the case with images, since there is a limited number of coefficients that can be watermarked in each image before the watermark is visible. Therefore, our video watermark detection algorithm calculated the number of watermarked DCT coefficients needed to compute the detector response to obtain the desired probability of detection, P_D for a given probability of a false alarm P_F. Our simulation results showed that the theoretically chosen value for N_w does lead to the desired P_D and P_F in Monte Carlo trials. Furthermore, even after watermarking attacks, we can obtain any P_D and P_F by making a worst case assumption on the first two parameters and computing N_w accordingly. We used this watermark detector to detect the watermark from I-frames of six standard video sequences watermarked with the algorithm presented in Chapter 3. Our simulation results showed that our watermarking scheme presented in Chapter 3 is robust to 3×3 Gaussian filtering, 50% cropping, addition of white noise $\mathcal{N}(0, 0.001)$, and a trivial deliberate attack.

In Chapter 5, we explored watermark embedding in P-frames [37, 40]. The challenge in embedding the watermark in P-frames is that the video bit rate increases significantly. Thus, we only embedded the watermark in nonzero quantized ac residuals in P-frames. Since these coefficients correspond to non-flat areas that are in motion, temporal and texture masking were exploited at the same time. We showed that the number of nonzero quantized ac residuals in each frame resembles the motion intensity plots in different video sequences. We also proposed embedding the watermark in nonzero quantized ac residuals with spatial masking capacity in I-frames. Our simulation results showed that the bit rate increase per watermark bit of our algorithm in this chapter is significantly smaller than the previous compressed-domain video watermarking algorithms [41, 76] when embedding the watermark in both I- and P- frames.

In some applications, the watermark detector does not know which coefficients are embedded with the watermark. For example, our watermark embedding algorithm in Chapter 5 embedded the watermark in the nonzero quantized ac residuals. The identity of those locations is lost when the video is completely decoded. Furthermore, a re-encoding or a deliberate attack that inserts additional nonzero coefficients would cause desynchronization and consequent failure in watermark detection. Thus, for some applications it is important that the watermark detection algorithm not depend on the precise location of the watermark signal. In Chapter 6, we proposed a new variation on the watermark detection algorithm developed in Chapter 4 that does not depend on where the watermark signal is embedded, called the location-unaware detector [37, 42]. Our algorithm calculated the number of frames, F, required for the watermark detector to obtain the desired probability of a detection, P_D, for a given false alarm probability, P_F. We used this detector to detect the watermark from P-frames of eight standard video sequences that were embedded with the watermark using our watermark embedding algorithm presented in Chapter 5. Our simulation results showed that the theoretically chosen value for F did lead to the desired values of P_D and P_F in Monte Carlo trials. We tested the robustness of our proposed algorithm to several common signal processing attacks such as filtering, 50% cropping, addition of white noise $\mathcal{N}(0, 0.001)$, and a requantization attack. Our simulation results showed that our proposed algorithm was robust against these attacks. We also compared the performance of location-unaware detector with the location-aware detector presented in Chapter 4. While both detectors could achieve any probability of detection and false alarm, the location-aware detector could generate detector responses more frequently than the location-unaware detector.

7.2 Suggestions for Future Work

There are many avenues for pursuing further work in this area. Some of those avenues are as follows:

- Exploring how far the Gaussian distribution model is from the optimal generalized Gaussian distribution model.
- Predicting how different attacks will affect $\hat{Q}$ and the variance of the DCT coefficients and adjusting the detector accordingly.
- Testing the robustness of the proposed algorithm to more severe attacks such as changing the video format.
- Exploring the watermarking capacity of chrominance components.
- Developing a watermarking scheme that avoids error propagation in I-frames when the watermark is embedded in the H.264 bitstream.
- Looking into security measures for watermarking that are similar to what exists in cryptography. For example, answering the questions: Is it possible to develop a security framework based on the computational complexity of attacks? Can this lead to a solid framework to evaluate and compare different watermarking schemes?

REFERENCES

[1] "H.264 reference software group." http://iphome.hhi.de/suehring/tml/, 2006.

[2] 14496-10, I. and H.264, I. R., "Advanced video coding," 2003.

[3] Alattar, A. M., Lin, E. T., and Celik, M. U., "Digital watermarking of low bit rate advanced simple profile MPEG-4 compressed video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 787–800, August 2003.

[4] ANSI (ANSI T1.801.03-2003), July 2003.

[5] Bender, W., Gruhl, D., and Morimoto, N., "Techniques for data hiding," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 2420, (San Jose, CA, USA), pp. 164–173, February 1995.

[6] Birney, K. A. and Fischer, T. R., "On the modeling of DCT and subband image data for compression," *IEEE Transactions on Image Processing*, vol. 4, pp. 186–193, February 1995.

[7] Boland, F. M., O'Ruanaidh, J. J. K., and Dautzenberg, C., "Watermarking digital images for copyright protection," in *Proceedings of International Conference on Image Processing and its Applications*, vol. 410, (Edinburgh, UK), pp. 326–330, July 1995.

[8] Bors, A. G. and Pitas, I., "Embedding parametric digital signatures in images," in *Proceedings EUSIPCO-96, VII European Signal Processing Conference*, vol. 3, (Trieste, Italy), pp. 1701–1704, September 1996.

[9] Bors, A. G. and Pitas, I., "Image watermarking using DCT domain constraints," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 3, (Lausanne, Switzerland), pp. 231–234, September 1996.

[10] Burgett, S., Koch, E., and Zhao, J., "Copyright labeling of digitized image data," *IEEE Communications Magazine*, vol. 36, pp. 94–100, March 1998.

[11] Caronni, G., "Assuring ownership rights for digital images," in *Proceedings of Reliable IT Systems (VIS)*, (Germany), 1995.

[12] Chen, B. and Wornell, G. W., "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Transactions on Information Theory*, vol. 47, pp. 1423–1443, May 2001.

[13] Cox, I. J., Kilian, J., Leighton, F. T., and Shamoon, T., "Secure spread spectrum watermarking for multimedia," *IEEE Transactions on Image Processing*, vol. 6, pp. 1673–1687, December 1997.

[14] Delaigle, J.-F., Vleeschouwer, C. D., Goffin, F., Macq, B., and Quisquater, J.-J., "Low cost watermarking based on a human visual model," in *Multimedia Applications, Services and Techniques - ECMAST'97. Second European Conference Proceedings*, (Milan, Italy), pp. 153–167, May 1997.

[15] Delaigle, J.-F., Vleeschouwer, C. D., and Macq, B., "Digital watermarking," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 2659, (San Jose, CA, USA), pp. 99–110, February 1996.

[16] Doerr, G. and Dugelay, J. L., "A guide tour of video watermarking," *Signal Processing: Image Communication*, vol. 18, pp. 263–282, April 2003.

[17] Hartung, F. and Girod, B., "Watermarking of uncompressed and compressed video," *Signal Processing*, vol. 66, pp. 283–301, May 1998.

[18] Hartung, F. and Kutter, M., "Multimedia watermarking techniques," *Proceedings of the IEEE*, vol. 87, pp. 1079–1107, July 1999.

[19] Hernandez, J. R., Amado, M., and Perez-Gonzalez, F., "DCT-domain watermarking techniques for still images: detector performance analysis and a new structure," *IEEE Transactions on Image Processing*, vol. 9, pp. 55–68, January 2000.

[20] Huang, X. and Zhang, B., "Robust detection of transform domain additive watermarks," in *Proceedings of Digital Watermarking 4th International Workshop (IWDW)*, vol. 3710, (Siena, Italy), pp. 124–138, September 2005.

[21] Inoue, H., Miyazaki, A., Yamamoto, A., and Katsura, T., "A digital watermark based on the wavelet transform and its robustness on image compression," in *Proceedings of International Conference on Image Processing (ICIP)*, vol. 2, (Chicago, IL, USA), pp. 391–395, October 1998.

[22] Jonsson, R. H., *Adaptive subband coding of video using probability distribution models*. PhD thesis, Georgia Institute of Technology, 1994.

[23] Koch, E., Rindfrey, J., and Zhao, J., "Copyright protection for multimedia data," in *Proceedings of the International Conference on Digital Media and Electronic Publishing*, (Leeds, UK), May 1994.

[24] Koch, E., Rindfrey, J., and Zhao, J., "Towards robust and hidden image copyright labeling," in *Proceedings of IEEE Workshop on Nonlinear Signal and Image Processing*, pp. 452–455, 1995.

[25] Kundur, D. and Hatzinakos, D., "A robust digital image watermarking method using wavelet-based fusion," in *Proceedings of International Conference on Image Processing (ICIP)*, vol. 1, (Santa Barbara, CA, USA), pp. 547–554, October 1997.

[26] Kutter, M., “Watermarking resisting to translation, rotation, and scaling,” in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 3528, pp. 423–431, 1998.

[27] Kutter, M., Jordan, F., and Bossen, F., “Digital signature of color images using amplitude modulation,” in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 3022, (San Jose, CA, USA), pp. 518–526, February 1997.

[28] Langelaar, G. C., Lagendijk, R. L., and Biemond, J., “Real-time labeling of MPEG2 compressed video,” *Journal of Visual Communication and Image Representation*, vol. 9, pp. 256–270, December 1998.

[29] Langelaar, G. C., van der Lubbe, J. C. A., and Biemond, J., “Copy protection for multimedia data based on labeling techniques,” in *Proceedings of 17th Symposium on Information Theory in the Benelux*, (Enschede, The Netherlands), May 1996.

[30] Langelaar, G. C., ven der Lubbe, J. C. A., and Lagendijk, R. L., “Robust labeling methods for copy protection of images,” in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 3022, (San Jose, CA, USA), pp. 298–309, February 1997.

[31] Liang, Y., Ahmad, I., and Swaminathan, V., “Fast priority search algorithm for block motion estimation,” in *IEEE International Conference on Multimedia and Expo (ICME)*, vol. 1, (Taipei, Taiwan), pp. 543–546, June 2004.

[32] Maes, M. J. J. B. and Overveld, C. W. A. M., “Digital watermarking by geometric wrapping,” in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 2, (Chicago, IL, USA), pp. 424–426, October 1998.

[33] Malvar, H. S., Hallapuro, A., Karczewicz, M., and Kerofsky, L., “Low-complexity transform and quantization in H.264/AVC,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 598–603, July 2003.

[34] Matsui, K. and Tanaka, K., “Video-steganography,” *Journal of the Interactive Multimedia Association Intellectual Property Project*, vol. 1, no. 1, pp. 187–205, 1994.

[35] Nikolaidis, A. and Pitas, I., “Asymptotically optimal detection for additive watermarking in the DCT and DWT domains,” *IEEE Transactions on Image Processing*, vol. 12, pp. 563–571, May 2003.

[36] Nikolaidis, N. and Pitas, I., “Copyright protection of images using robust digital signatures,” in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing Conference (ICASSP)*, no. 4, (Atlanta, GA, USA), pp. 2168–2171, May 1996.

[37] Noorkami, M. and Mersereau, R. M., "Digital video watermarking in P-frames with controlled video bit rate increase," *IEEE Transactions on Information Forensics and Security*. Submitted.

[38] Noorkami, M. and Mersereau, R. M., "Compressed-domain video watermarking for H.264," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 2, (Genoa, Italy), pp. 890–893, September 2005.

[39] Noorkami, M. and Mersereau, R. M., "Towards robust compressed-domain video watermarking for H.264," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 6072, (San Jose, CA, USA), January 2006.

[40] Noorkami, M. and Mersereau, R. M., "Digital video watermarking in P-frames," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 6505, (San Jose, CA, USA), January 2007.

[41] Noorkami, M. and Mersereau, R. M., "A framework for robust watermarking of H.264-encoded video with controllable detection performance," *IEEE Transactions on Information Forensics and Security*, vol. 2, pp. 14–23, March 2007.

[42] Noorkami, M. and Mersereau, R. M., "Video watermark detection with and without knowledge of watermark location," in *8th ACM Multimedia and Security Workshop*, (Dallas, TX, USA), September 2007. Submitted.

[43] O'Ruanaidh, J. J. K., Dowling, W. J., and Boland, F. M., "Phase watermarking of digital images," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 3, (Lausanne, Switzerland), pp. 239–242, September 1996.

[44] O'Ruanaidh, J. J. K. and Pun, T., "Rotation, scale and translation invariant digital image watermarking," in *Proceedings of International Conference on Image Processing (ICIP)*, vol. 1, (Santa Barbara, CA, USA), pp. 536–539, October 1997.

[45] Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., and Wedi, T., "Video coding with H.264/AVC: Tools, performance, and complexity," *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7–28, 2004.

[46] Peterson, H. A., Ahumada, A. J., and Watson, A. B., "An improved detection model for DCT coefficient quantization," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 1913, (San Jose, CA, USA), pp. 191–201, February 1993.

[47] Peterson, H. A., Peng, H., Morgan, J. H., and Pennebaker, W. B., "Quantization of color image components in the DCT domain," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 1453, pp. 210–222, 1991.

[48] Pitas, I., "A method for signature casting on digital images," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 3, (Lausanne, Switzerland), pp. 215–218, September 1996.

[49] Pitas, I., "A method for watermark casting on digital images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 775–780, October 1998.

[50] Piva, A., Barni, M., Bartolini, F., and Cappellini, V., "DCT-based watermark recovering without resorting to the uncorrupted original image," in *Proceedings of International Conference on Image Processing (ICIP)*, vol. 1, (Santa Barbara, CA, USA), pp. 520–523, October 1997.

[51] Podilchuk, C. I. and Zeng, W., "Perceptual watermarking of still images," in *Proceedings of Signal Processing Society Workshop on Multimedia Signal Processing*, pp. 363–368, 1997.

[52] Podilchuk, C. I. and Zeng, W., "Digital image watermarking using visual models," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 3016, (San Jose, CA, USA), pp. 100–111, February 1997.

[53] Puate, J. and Jordan, F., "Using fractal compression scheme to embed a digital signature into an image," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 2915, (Boston, MA, USA), pp. 108–118, November 1996.

[54] Qiu, G., Marziliano, P., Ho, A. T. S., He, D., and Sun, Q., "A hybrid watermarking scheme for H.264/AVC video," in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 4, (Cambridge, UK), pp. 865–868, August 2004.

[55] Richardson, I. E. G., *H.264 and MPEG-4 Video Compression*. Wiley, 2004.

[56] Sayood, K., *Introduction to Data Compression*, ch. 13. New York: Morgan Kaufmann, 1996.

[57] Simitopoulos, D., Tsaftaris, S. A., Boulgouris, N. V., and Strintzis, M. G., "Compressed-domain video watermarking of MPEG streams," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, vol. 1, (Lausanne, Switzerland), pp. 569–572, August 2002.

[58] Smith, J. R. and Comiskey, B. O., "Modulation and information hiding in images," in *Proceedings of Information Hiding International Workshop*, (Cambridge, UK), pp. 207–226, May 1996.

[59] Su, K., Kundur, D., and Hatzinakos, D., "Spatially localized image-dependent watermarking for statistical invisibility and collusion resistance," *IEEE Transactions on Multimedia*, vol. 7, pp. 52–66, February 2005.

[60] Su, K., Kundur, D., and Hatzinakos, D., "Statistical invisibility for collusion-resistant digital video watermarking," *IEEE Transactions on Multimedia*, vol. 7, pp. 43–51, February 2005.

[61] Swanson, M. D., Zhu, B., and Tewfik, A. H., "Robust data hiding for images," in *Proceedings of IEEE Digital Signal Processing Workshop*, (Loen, Norway), pp. 37–40, September 1996.

[62] Swanson, M. D., Zhu, B., and Tewfik, A. H., "Transparent robust image watermarking," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 3, (Lausanne, Switzerland), pp. 211–214, September 1996.

[63] Swanson, M. D., Zhu, B., and Tewfik, A. H., "Multiresolution scene-based video watermarking using perceptual models," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 540–550, May 1998.

[64] Tanaka, K., Nakamura, Y., and Matsui, K., "Embedding secret information into a dithered multi-level image," in *Proceedings of IEEE Military Communications Conference*, vol. 1, (Monterey, CA, USA), pp. 216–220, 1990.

[65] Tanaka, K., Nakamura, Y., and Matsui, K., "Embedding the attribute information into a dithered image," in *Systems and Computers in Japan*, vol. 21, pp. 43–50, 1990.

[66] Tao, B. and Dickinson, B., "Adaptive watermarking in the DCT domain," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, (Munich, Germany), pp. 2985–2988, April 1997.

[67] Tirkel, A. Z., Rankin, G. A., van Schyndel, R. M., Ho, W. J., Mee, N. R. A., and Osborne, C. F., "Electronic water mark," in *Conference Proceedings of Digital Image Computing: Techniques and Applications (DICTA)*, vol. 2, (Sydney, NSW, Australia), pp. 666–673, December 1993.

[68] Tirkel, A. Z., van Schyndel, R. G., and Osborne, C. F., "A two-dimensional digital watermark," in *Conference Proceedings Digital Image Computing: Techniques and Applications (DICTA)*, (Brisbane, Qld., Australia), pp. 378–383, December 1995.

[69] Trappe, W., Wu, M., and Liu, K. R., "Collusion-resistant fingerprinting for multimedia," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, (Orlando, FL, USA), pp. IV3309–3312, May 2002.

[70] van Schyndel, R. G., tirkel, A. Z., and Osborne, C. F., "A digital watermark," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 2, (Austin, TX, USA), pp. 86–89, November 1994.

[71] Van Trees, H. L., *Detection, Estimation, and Modulation Theory Part I.* New York: John Wiley and Sons, 1968.

[72] Watson, A. B., "DCT quantization matrices visually optimized for individual images," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 1913, (San Jose, CA, USA), pp. 202–216, February 1993.

[73] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A., "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 560–576, July 2003.

[74] Wolfgang, R. B. and Delp, E. J., "A watermark for digital images," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 3, (Lausanne, Switzerland), pp. 219–222, September 1996.

[75] Wolfgang, R. B. and Delp, E. J., "A watermarking technique for digital imagery: Further studies," in *Proceedings of the International Conference on Imaging Science, Systems, and Technology (CISST)*, vol. 1, (Las Vegas, NV, USA), pp. 279–287, June 1997.

[76] Wolfgang, R. B., Podilchuk, C. I., and Delp, E. J., "Perceptual watermarks for digital images and video," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 3567, (San Jose, CA, USA), pp. 40–51, January 1999.

[77] Wu, G.-Z., Wang, Y.-J., and Hsu, W.-H., "Robust watermark embedding/detection algorithm for H.264," *Journal of Electronic Imaging*, vol. 14, pp. 13013–1–9, January 2005.

[78] Wu, M., Miller, M. L., Bloom, J. A., and Cox, I. J., "A rotation, scale and translation resilient public watermark," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, (Phoenix, AZ, USA), p. 2065, March 1999.

[79] Xia, X., Boncelet, C. G., and Arce, G. R., "A multiresolution watermark for digital images," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 1, (Santa Barbara, CA, USA), pp. 548–551, October 1997.

[80] Zeng, W. and Liu, B., "A statistical watermark detection technique without using original images for resolving rightful ownerships of digital images," *IEEE Transactions on Image Processing*, vol. 8, pp. 1534–1548, November 1999.

[81] Zhu, W., Xiong, Z., and Zhang, Y.-Q., "Multiresolution watermarking for images and video: a unified approach," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 1, (Chicago, IL, USA), pp. 465–468, October 1998.

VITA

Maneli Noorkami received the B.S. degree in Electrical Engineering from Sharif University, Tehran, Iran, in 2001 and the M.S. degree in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, GA, in 2003, where her Master's thesis focused on cryptography. She received her Ph.D. from the School of Electrical and Computer Engineering at Georgia Institute of Technology in 2007. Her research interests are in image and video processing with an emphasis on watermarking techniques for multimedia. She received the Outstanding Research Award from the Center for Signal and Image Processing at the Georgia Institute of Technology in 2006. She spent two internships at the R&D group in Texas Instrument Incorporated, Dallas, TX, developing signal processing software for image and video applications.