WATERMARK AND DATA HIDING EVALUATION: THE DEVELOPMENT OF A STATISTICAL ANALYSIS FRAMEWORK

A Thesis

Submitted to the Faculty

of

Purdue University

by

Hyung Cook Kim

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2006

Purdue University

West Lafayette, Indiana

To my parents Jong Yul Kim and Ye Hwan Ko

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Edward J. Delp, for his helpful advice, invaluable guidance, and teaching the virtues of "multitasking" which I hope I learn some day. He has provided many opportunities to explore the most interesting and challenging pursuits, and his teachings in scholarly research and in life shall always be cherished and remembered. He has been an advisor of the highest regard.

I would also like to thank my graduate committee: Professors Leah H. Jamieson, Michael D. Zoltowski, and Zygmunt Pizlo, for their advice, encouragement, criticism, and insight. They have encouraged me to look deeper, to explore and discover new perspectives.

I would like to especially thank the Air Force Research Laboratory, Information Directorate, Rome Research Site, for funding the research in this dissertation. Without their support, this dissertation would not have been possible.

I appreciate the support and friendship of all my colleagues in the Video and Image Processing (VIPER) lab, past and present: Dr. Gregory Cook, Dr. Cuneyt Taskiran, Dr. Eduardo Asbun, Dr. Eugene T. Lin, Dr. Yuxin Liu, Dr. Jinwha Yang, Dr. Sahng-Gyu Park, Dr. Yajie Sun, Dr. Lauren Christopher, Dr. Zhen Li, Anthony Martone, Aravind Mikkilineni, Hwayoung Um, Limin Liu, Jennifer Talavage, Michael Igarta, Oriol Guitart, Hakeem Ogunleye, and Ashok Mariappan. This journey would not have been as special, or as memorable, without their friendship. I would like to thank my colleagues from abroad for their encouragement and perspective: Professor Josep Prades-Nebot, Rafael Villoria, Andreas Lang, and Professor Hwanil Kang.

Finally, I would like to thank my family for their support and encouragement. My father has encouraged me to try my best, be organized, and exercise because you have to be prepared for the future. My mother always gave me a new perspective, reminded me of things I should not forget, and worried about my health which is pretty good. I also would like to thank my sister for being more mature than me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Kim, Hyung Cook. Ph.D., Purdue University, May, 2006. Watermark and Data Hiding Evaluation: The Development of a Statistical Analysis Framework. Major Professor: Edward J. Delp.

Digital watermarking is the practice of hiding a message in an image, audio, video or other digital media elements. Since the late 1990's, there has been an explosion in the number of digital watermarking algorithms published. But there are few widely accepted tools and metrics that can be used to validate the performance claims being asserted by members of the research community.

Robust image watermarks are watermarks designed to survive attacks including signal processing operations and spatial transformations. To evaluate robust watermarks, we need to evaluate how attacks affect the fidelity of an image. The mean square error (MSE) is the most popular metric to measure fidelity. MSE, as it is, cannot measure the fidelity for images that went through geometric attacks such as rotation, pixel loss attacks such as cropping, or valumetric attacks such as gamma correction. We take the approach of evaluating attacks using MSE by compensating valumetric, pixel loss, and geometric attacks using conditional mean, error concealment, and motion compensation, respectively.

Robust watermarks are evaluated in terms of fidelity and robustness. To measure robustness, bit error rate, message error rate and the receiver operating characteristic (ROC) of the watermark decoder and detector are currently used in the literature. We extend this framework by adopting reliability testing. We define reliability as the probability that a watermarking algorithm will correctly detect or decode a watermark for a specified fidelity requirement under a given set of attacks and images. We evaluate three watermarking algorithms in terms of quality (fidelity), load (watermark strength, payload), capacity (minimum watermark strength, maximum payload), and performance (robustness). We adopt the Taguchi loss function which is a compromise between average and percentile to summarize fidelity and performance.

To facilitate the use of a watermark evaluation method, we need a watermark evaluation benchmark that implements that method. To meet this need, we developed the watermark evaluation testbed (*WET*). This system consists of reference software that includes both watermark embedders and detectors, attacks, evaluation modules and image database.

1. INTRODUCTION

In order to prevent forgery in checks and bank notes, paper watermarks are commonly used. Analogous to paper watermarks, digital watermarking is the practice of hiding a message in an image, audio, video or other digital media elements. Since the late 1990s, there has been an explosion in the number of digital watermarking algorithms published [1–6]. That is most likely due to the increase in concern over copyright protection of content [7]. Because digital devices store content in digital form, there is no degradation in quality of data [8]. The popularization of Internet and digital recording devices caused piracy to increase and content providers are seeking technologies to protect their rights [9]. Currently, cryptographic techniques are the most popular method to prevent piracy [10]. But after the content is decrypted, we need another way to protect the content. Because digital watermarking can embed information related to the content in the digital media element and can be designed to survive many changes such as format conversion, D/A and A/D conversion and compression, it is seen as a technique to complement cryptography in preventing piracy [11].

1.1 Watermarking Applications

While copyright protection is usually the major driving force in the watermark field, there are a number of possible applications of watermarking. These [5, 12–14] include:

- Owner Identification: If an owner of a copyrighted material does not put a copyright notice on the distributed material, the award for the copyright holder can be limited in case the material has not been used as agreed upon. The problem with copyright notices is that they can be easily moved and may cover a portion of the material. Because watermarks can be made both imperceptible and inseparable from the material that contains them, they can be a better alternative to copyright notices [15].

- Proof of Ownership: Proof of ownership is using watermarks to provide proof of ownership in a court of law. For a watermarking algorithm to be used in a proof of ownership application, it should not be susceptible to the ambiguity attack described later in this chapter [6, 16].

- Transaction Tracking: In transaction tracking, or fingerprinting, a unique watermark is embedded into each copy of a media element. The embedded watermark will be present in any copy of the watermarked signal. The watermark identifies the legal recipient of the copy and can be used to trace the source of copied content.

- Copy Control: In the copy control application, we aim to prevent people from making illegal copies of copyrighted content. Current technologies that do this is encryption and Macrovision's Videocassette Anticopy Process. These technologies do not survive format changes of the content. Because watermarks are embedded in the content itself, they are present in every variation of the content and therefore might provide a better method of implementing copy control.

- Content Authentication [17–20]: It is becoming easier to tamper with digital content in ways that are difficult to detect. One common cryptographic approach for authenticating messages is using digital signature, which is an encrypted summary of the image. The problem with digital signature is that they can be lost during normal usage. By using watermarking to embed the signature in the content, we can make the signature stay with the media element. An image with a missing or damaged watermark could imply some kind of tampering has occured and also indicate where the tampering has taken place.

- Device Control: There are several applications in which devices react to watermarks they detect in content. From the user's point of view, many of these are different from copy control in that they add value to the content, rather than restrict its use. These include linking toys with television or websites with images.

- Broadcast Monitoring: We can use watermarks for broadcast monitoring by putting a unique watermark in each video or sound clip prior to broadcast. Automated monitoring stations can then receive broadcasts and look for these watermarks, identifying when and where each clip appears.

- Steganography: Steganography is the hiding of a secret message within an ordinary medial element and the extraction of it at its destination. Steganography takes cryptography a step farther by hiding an encrypted message so that no one suspects it exists [21]. Steganography can be used to send messages to other people without being detected.

1.2 Watermarking Overview

The objectives of digital watermarking is to embed or insert a message into a signal in a secure and imperceptible manner and to detect the embedded information from a watermarked signal. A watermarked image may be attacked, such as JPEG compression, before it is available to the watermark detector. A block diagram of a typical watermarking system [4] is shown in Figure 1.1. In the following we will explain watermarking techniques that currently exist in the literature. Although the techniques described can be applied to other medial elements, we will mainly focus on still images.

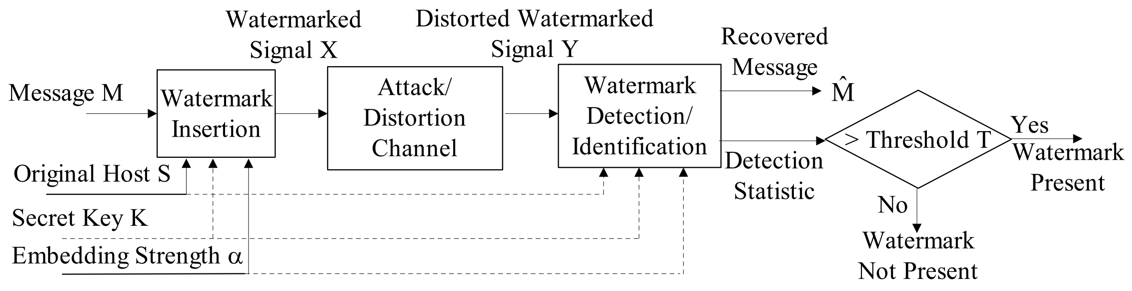


Fig. 1.1. A block diagram of a typical watermarking system.

1.2.1 Watermark Embedding

Watermark embedding is hiding a message into an image by mapping the message to another signal and adding that signal to an image. That signal is also called the watermark. Components of an embedder are shown in Figure 1.1. A watermark embedder accepts as inputs the host image S, secret key K, and message M and produces the watermarked signal X. The embedding key K is the secret that is necessary to detect the watermark and decode the message in the watermark. The embedding strength parameter α is a parameter to control the energy of the watermarked signal. As the signal strength of the watermark increases, usually the visibility of the watermark and the probability to detect or decode the watermark increases. A secret key K or key pair is used for watermark embedding and detection. This key is analogous to the secret PN-sequences (chips) used in spread spectrum communications. Although, the size of the watermark key space has no direct impact on some properties of watermark such as fidelity and robustness, it plays an important role in the security of the system. The key space, that is the range of all possible values of the watermark keys, must be large enough to make exhaustive search attacks impossible.

Marking Space

Although, watermarking can be done in the spatial domain, it is usually done in the transform domain to spread the watermark all over the image [1].

Discrete Fourier Transform The unitary Discrete Fourier Transform (DFT) of a sequence $f(n), n = 0, \ldots, N-1$ is defined [22] as

$$F(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n) \exp^{-j\frac{2\pi nk}{N}}, \quad k = 0, \ldots, N-1 \tag{1.1}$$

The inverse transform is given by

$$f(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) \exp^{j\frac{2\pi kn}{N}}, \quad n = 0, \ldots, N-1 \tag{1.2}$$

The translational property and reciprocal scaling property of DFT make the DFT a good candidate for developing watermarks that are robust to translation and scaling [23].

Discrete Cosine Transform The one-dimensional Discrete Cosine Transform (DCT) of a sequence $f(n), 0 \leq n \leq N-1$ is defined as [22]

$$F(k) = \alpha(k) \sum_{n=0}^{N-1} f(n) \cos[\frac{\pi(2n+1)k}{2N}], \quad 0 \leq k \leq N-1 \tag{1.3}$$

where

$$\alpha(0) = \sqrt{\frac{1}{N}}, \quad \alpha(k) = \sqrt{\frac{2}{N}} \quad \text{for } 1 \leq k \leq N-1 \tag{1.4}$$

DCT has been widely studied by the source coding community in the context of JPEG and MPEG [24, 25]. The main arguments for using DCT in watermarking are the following. Watermarking algorithms that work in the DCT domain are often more robust to JPEG and MPEG compression. Furthermore, studies on visibility (i.e., visual distortions) which were previously conducted in the field of source coding can be reused [26] to hide the watermark more effectively to the human visual system (HVS). Also, watermarking in the DCT domain offers the possibility of combining compression and watermarking (i.e., inside a JPEG or MPEG encoder) in order to minimize the computation time.

Discrete Wavelet Transform Discrete Wavelet Transform (DWT) is used in the source compression standard JPEG 2000. In several recent publications, this transform has been used in image watermarking. The reason for using DWT closely follows those for using DCT (i.e. preventing watermark removal by JPEG 2000 lossy compression, reusing previous studies on source coding regarding the visibility of image degradations, and offering the possibility of embedding in the compressed domain). In addition to these criteria, the multiresolution aspect of wavelets is helpful in spreading the watermark in regions where the watermark is less visible than other regions.

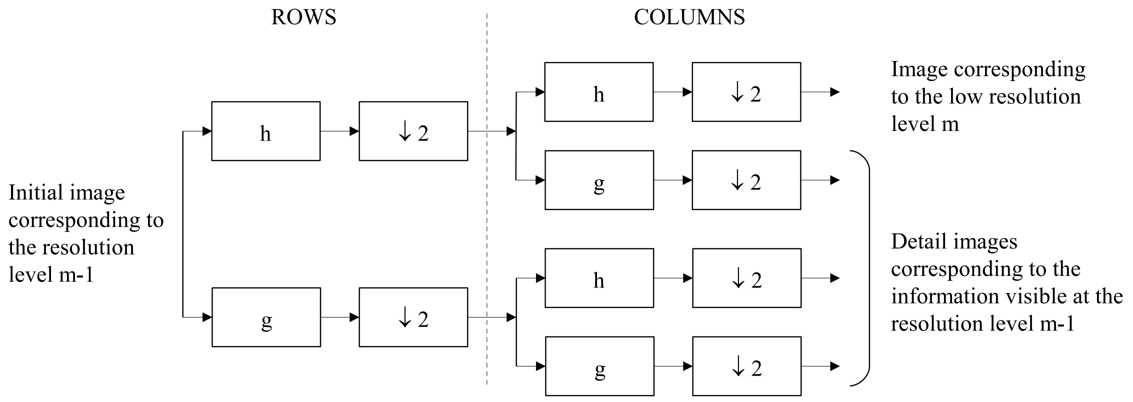
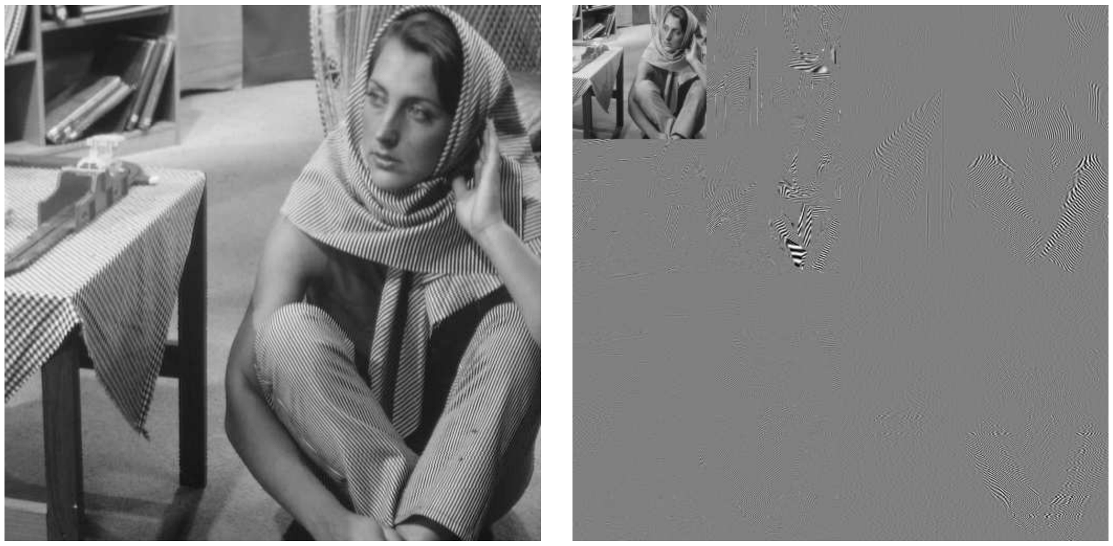

ROWS
COLUMNS
Initial image corresponding to the resolution level m-1
h
g
↓2
Image corresponding to the low resolution level m
Detail images corresponding to the information visible at the resolution level m-1

Fig. 1.2. A block diagram of a two-dimensional discrete wavelet transform (DWT).

The basic idea of the Discrete Wavelet Transform (DWT) is the following. A signal is split into two parts, usually high frequencies and low frequencies. The edge components of the signal are largely confined in the high frequency part. The low frequency part is split again into two parts of high and low frequency. For two-dimensional images, we apply the DWT in the row and column directions as shown in Figure 1.2. It is desirable that we use the linear phase finite impulse response (FIR) filters, since such filters can be easily cascaded in pyramid filter structures without the need for phase compensation [27]. One can preserve linear phase by relaxing the orthonormality requirement, and using biorthogonal bases. An example of a two-level wavelet decomposition using the Daubechies 9x7 filter [27] is shown in Figure 1.3.

Watermark Generation

Many watermark algorithms use a sequence with each element having a normal distribution $N(0,1)$ as the watermark. This makes the watermark resilient to collu-



(a) Original image: Barbara(512x512) (b) 2 level wavelet decomposition

Fig. 1.3. An example of a two-level wavelet decomposition using the Daubechies 9x7 filter.

sion attacks [1]. One way to generate normal random variables is to use the polar method from [28]. Let

$$\Theta \sim U(0,1) \quad \text{and} \quad S \sim U(0,1). \tag{1.5}$$

Set X_1 and X_2 as follows:

$$X_1 = \sqrt{-2\ln S}\cos 2\pi\Theta \quad \text{and} \quad X_2 = \sqrt{-2\ln S}\sin 2\pi\Theta. \tag{1.6}$$

Then X_1 and X_2 are independent with mean 0 and standard deviation 1. The algorithm in [28] avoids the use of cosine and sine.

Embedder Classification

During watermark embedding, the host signal is known to the embedder and can be treated as side information about the distortion channel rather than as unknown noise. Based on how we use this information, we can categorize watermarking embedding as follows [5].

Blind Embedder Blind Embedder is an embedder that ignores the host signal, shown in Figure 1.4. Many blind embedders [29] are of the form

$$\mathbf{X} = \mathbf{S}+\alpha\mathbf{W}_m \tag{1.7}$$

, where $\mathbf{X}$ is the watermarked signal, $\mathbf{S}$ is the original signal, $\mathbf{W}_m$ is a message mark which depends only on the key and message for blind embedders and α is a scaling constant.

Informed Embedder Informed embedder is an embedder that produces a watermark depending on the characteristics of the host image as shown in Figure 1.5. One example of watermarking techniques with informed embedding is perceptually adaptive watermarking techniques [3,30,31]. A watermarking technique can exploit the characteristics of a Human Visual System (HVS) , such as masking, sensitivity and pooling [32, 33], to produce an imperceptible watermarked signal by using a perceptual model [26, 31, 33–35].

Another example of informed embedding is the Secure Spread Spectrum Watermarking Algorithm by Cox et al. [1]. The algorithm is based on the assumption that a watermark should be placed in the perceptually most significant components of data. For imperceptibility, the algorithm spreads the watermark among the significant components of data. It watermarks an $N \times N$ image by computing the $N \times N$ 2-dimensional(2D) DCT(Discrete Cosine Transform) of the image and placing the watermark into the n highest magnitude coefficients of the transform matrix, excluding the DC component. We obtain the watermarked image by applying inverse DCT to the $N \times N$ DCT coefficients. The algorithm extracts the n highest magnitude DCT coefficients $S = s_1, \ldots, s_n$ and inserts a watermark $W = w_1, \ldots, w_n$ to obtain an adjusted sequence of values $X = x_1, \ldots, x_n$. X is inserted back into the 2D DCT coefficients. The DCT coefficients are watermarked as follows:

$$x_i = s_i(1 + \alpha w_i) \tag{1.8}$$

, where α is the embedding strength. The watermark is detected by comparing the extracted watermark W^* and the original watermark W. The similarity of W and W^* is measured by a scaled version of normalized correlation

$$sim(W, W^*) = \frac{W^* \cdot W}{\sqrt{W^* \cdot W^*}}. \tag{1.9}$$

A watermarked image and its watermark for the DCT spread spectrum watermarking are shown in Figure 1.6 for embedding strength $\alpha = 1$ and watermark length $n = 1000$. Instead of DCT we can use DWT. Watermarked image and its watermark for the DWT spread spectrum watermarking is shown in Figure 1.7 for embedding strength $\alpha = 0.1$, watermark length $n = 1000$, and wavelet decomposition level of 9. As we can see from the difference images, wavelet watermarks is more localized in a specific area.

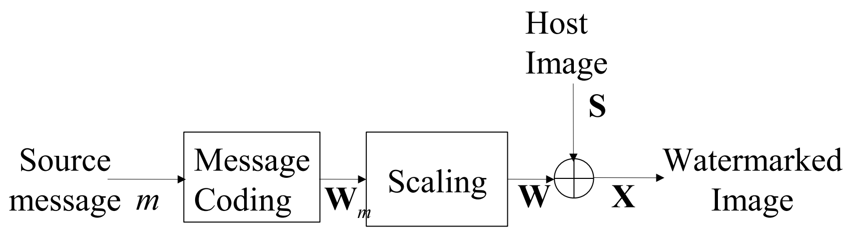


Fig. 1.4. A block diagram of a blind embedder that ignores the host image.

Informed Coder and Embedder Informed coding refers to the use of side information during the selection of a message mark $\mathbf{W}_m$. This can be done by defining a dirty-paper code [36] in which message is represented by several alternate vectors. We then select the watermark that represents the desired message and is closest to the host image. By combining informed coding and informed embedding as shown in Figure 1.8, it is possible to obtain much better performance than with simple blind coding and embedding. In fact, there is reason to believe that the amount of information we can reliably embed might be independent of the distribution of unwatermarked content. Informed embedding and informed coding are also called watermarking with side information [37]. An example of informed coding and informed embedding combined is Quantization Index Modulation technique by Chen et al. [38,39]. Quantization Index Modulation is a method of watermarking in which each message is associated with a distinct vector quantizer. The embedder quantizes the host signal (or a vector extracted from the host signal) according to the quantizer associated with the desired message. The detector quantizes the image using the union of all quantizers, and identifies the message.

1.2.2 Watermark Detection and Decoding

The watermark detector decides whether a watermark is present in an image. The decision is usually made by comparing a detection statistic with a threshold T. The detection statistic is any measure of the likelihood that a watermark is present in

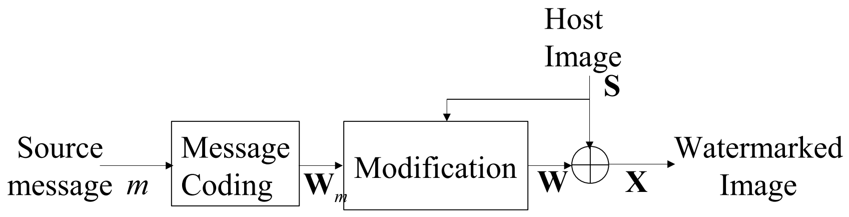
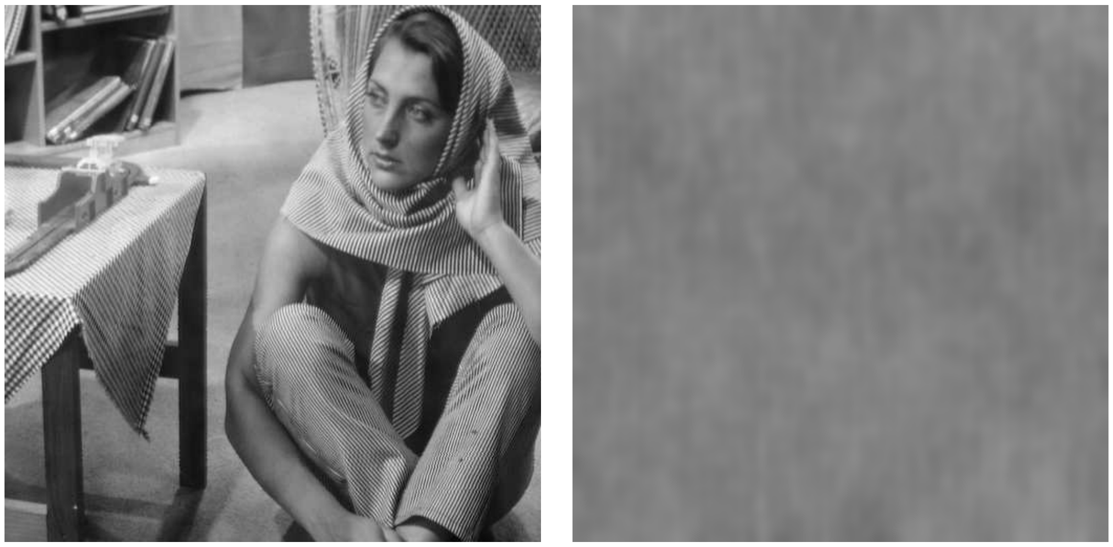
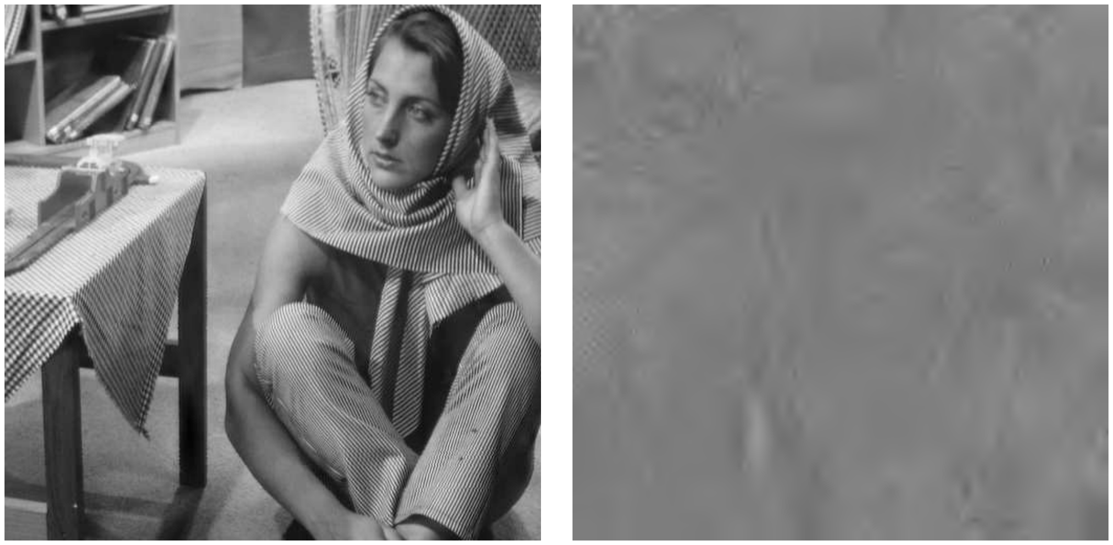


Fig. 1.5. A block diagram of an informed embedder that produces a watermark depending on the characteristics of the host image.



(a) Watermarked image (MSE = 23.0) (b) Watermark added to the image

Fig. 1.6. A watermarked image (a) and its watermark (b) for a DCT spread spectrum watermarking.



(a) Watermarked image (MSE=22.3) (b) Watermark added to the image

Fig. 1.7. A watermarked image (a) and its watermark (b) for a DWT spread spectrum watermarking.

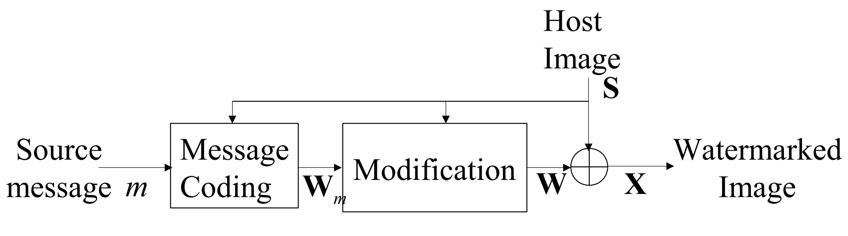


Fig. 1.8. A watermark block diagram using informed coding and embedding.

the image. The value of T depends on the detection probability requirements. For detection statistics, majority of the algorithms use similarity measures [5]. There are also algorithms that use maximum likelihood (ML) and maximum a posteriori probability (MAP) estimates [40]. Common similarity measures include linear correlation, normalized correlation, and the correlation coefficient. Watermark decoder maps the watermarked image into a message. Watermark decoders also use similarity measures, ML, and MAP estimates to determine the message.

A detector that requires access to the original, unwatermarked image is an informed detector. This term also refer to detectors that require some information derived from the original image, rather than the entire image. Conversely, detectors that do not require any information related to the original are referred to as blind detectors.

Similarity Measures

For watermark detection in correlation-based watermarking systems, linear correlation, normalized correlation and correlation coefficient values are usually compared against a threshold.

Linear correlation of two N-dimensional vectors $\mathbf{v}$ and $\mathbf{w_r}$ is defined as

$$z_{lc}(\mathbf{v}, \mathbf{w_r}) = \frac{1}{N}\mathbf{v} \cdot \mathbf{w_r} \tag{1.10}$$

It is common in practice in communications to test for the presence of a transmitted signal, $\mathbf{w_r}$, in a received signal, $\mathbf{v}$, by computing linear correlation and comparing it to a threshold. The practice is referred to as matched filtering and is known to be an optimal method of detecting signals in the presence of additive, white Gaussian noise [41].

One of the problems with linear correlation is that it is vulnerable to amplitude scaling. To solve this, normalized correlation of two vectors, $\mathbf{w_r}$ and $\mathbf{v}$ defined as

$$z_{lc}(\mathbf{v}, \mathbf{w_r}) = \frac{\mathbf{v} \cdot \mathbf{w_r}}{|\mathbf{v}||\mathbf{w_r}|} \tag{1.11}$$

is used.

Correlation coefficient is a form of normalized correlation in which the vector is first modified to have zero mean. This provides robustness against changes in the DC term of an image, such as the addition of a constant intensity to all pixels of an image. Correlation coefficient is defined as

$$\begin{aligned} \tilde{\mathbf{v}} &= \mathbf{v} - \bar{\mathbf{v}} \\ \tilde{\mathbf{v}} &= \mathbf{v} - \bar{\mathbf{v}} \\ z_{cc}(\mathbf{v}, \mathbf{w_r}) &= z_{nc}(\tilde{\mathbf{v}}, \tilde{\mathbf{v}}). \end{aligned} \tag{1.12}$$

1.2.3 Attacks

To test the robustness and security properties of watermarks, we have to use various attacks that models the processes the media element will go through based on the application. We describe attacks and counter-attacks that are from [5,21,42–44].

Categories of Attack

The wide class of existing attacks can be divided into four main categories: interference and elimination attacks, presentation attacks, cryptographic attacks and protocol attacks. Figure 1.9 summarizes the different attacks.

Interference and Elimination Attacks Elimination attacks [45] aim at complete removal of a watermark from host data. The interference attacks are those which add noise to the watermarked image.

- Compression: JPEG [24] is currently one of the most widely used lossy compression algorithm for images. Any still image watermarking system should be resilient to some degree of lossy compression because images are usually lossy compressed to reduce its file size significantly. Many people have rec-

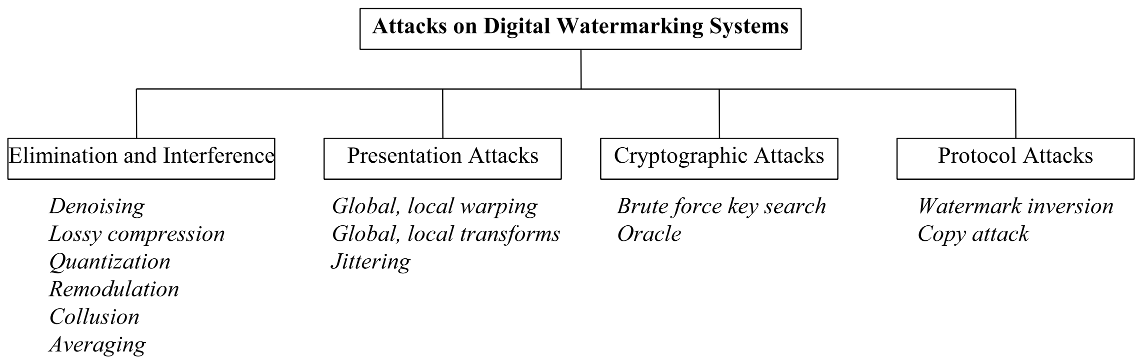


Fig. 1.9. Types of attacks on digital watermarking systems.

ognized that there is a fundamental conflict between watermarking and lossy compression.

- Noise addition: Various types of noise are added to the image by the imaging system [46]. Additive noise and uncorrelated multiplicative noise have been largely addressed in the communication theory and signal processing theory literature.

- Statistical averaging and collusion: Given two or more copies of the same image but with different watermarks, the watermarks should survive when the images are averaged or a new image is constructed by taking different parts from each image. Collusion attacks are applicable when many copies of a given data set, each signed with a key or different watermark, can be obtained by an attacker or a group of attackers. In such a case, a successful attack can be achieved by averaging all copies.

- Low pass filtering: This includes linear and non-linear filters. Frequently used filters include Gaussian, and standard average filters. Since natural images generally have energy concentrated in the low frequency components, low pass filtering are sometimes used to reduce noise in an image.

- Sharpening: Sharpening functions are used to enhance edges. These filters can be used as a tool to actually see the watermark embedded in an image when the watermark are embedded in the high frequency components to reduce watermark visibility.

- Color quantization: This is mostly applied when the output devices can only represent limited number of colors. Color quantization is very often accompanied by dithering which diffuses the error of the quantization.

- Restoration: These techniques are usually designed to reduce the effects of specific degradation processes but could also be used without a priori knowledge of the noise introduced by the watermarking system.

- Denoising: The basic idea of this approach consists in the assumption that the watermark is noise which can be modeled statistically. Therefore, estimating the host image from a watermarked image, an attacker can potentially remove the watermark. Image denoising is mostly based on a maximum likelihood (ML), a maximum a posteriori probability (MAP), a minimum mean square error (MMSE) or a minimax criteria. In the case of the ML, the well-known denoising algorithms are local mean, median, trimmed mean and myriad filter [47] which are the estimates for a Gaussian, Laplacian, ε-contaminated (mixture model of a Gaussian and Laplacian), and Cauchy watermark distributions, respectively. The representatives of the MAP-estimates are the adaptive Wiener filter, soft and hard shrinkage [48].

Presentation Attacks Presentation attacks make the watermark undetectable by existing detectors without removing the watermark [49, 50].

- Amplitude changes: Amplitude changes for music means change of volume. In images and video, it represents a change in brightness and contrast. Linear correlation detection does not cope well with amplitude scaling.

- Gamma correction: Gamma correction is frequently used to enhance images or adapt images for display.

- Vertical, Horizontal, and Diagonal Flip: Many images can be flipped without losing any value. Some watermarking detectors consider all the flips and detect watermarks for each case.

- Rotation: Small angle rotation, often in combination with cropping, does not usually change the commercial value of the image but can make the watermark undetectable. Rotations are used to realign horizontal features of an image and it is one of the modifications applied to an image after it has been scanned.

- Cropping: From a large image, people crop a part of the image they want. Web sites sometimes divide images into several pieces. An example of cropping is the mosaic attack [51].
- Scaling: This happens when a printed image is scanned or when a high resolution image from a digital camera is scaled down to send by email or to post it on a Web site. Scaling can be divided into two groups, uniform and non-uniform scaling. Uniform scaling preserves the aspect ratio of the image while non-uniform scaling does not.
- Row or column removal: In interlaced video, every odd or even line is removed from each frame. Row or column removal attack is not easily noticed by the human visual system. Randomly removing columns or rows is also called a jitter attack.
- Generalized geometrical transformations: A generalized geometrical transformation is a combination of non-uniform scaling, rotation, and shearing.
- Local random bending: This attack locally and imperceptibly warps an image. This is also called the StirMark [51, 52] attack.

Cryptographic Attacks Cryptographic attacks are very similar to the attacks used in cryptography. Cryptographic attacks aim at breaking the security methods in watermarking schemes. They can remove a watermark, read the embedded message and add a watermark in an image with a different message.

- Brute-force search: Brute-force search is trying all the keys to get the key used to embed the watermark. The correct key can be found from the detection statistic or from the decoded message if we have a prior knowledge of the message embedded.

- Oracle attack: When a watermark detector is available that tells you whether a watermark is embedded or not, an attacker can remove a watermark by applying small changes to the image until the watermark is detected.

Protocol Attacks Protocol attacks are attacks that prevent the watermark from doing its intended function in a watermarking applications [5].

- Over-marking: It is embedding another watermark to a media element. In this case the attacker needs special access to the watermarking software. Current commercial implementations prevent this by refusing to add a watermark if another is already embedded.

- Ambiguity attacks: Ambiguity attacks is based on inversion. The idea behind inversion is that the attacker subtracts his own watermark from watermarked data and claims to be the owner of the watermarked data. This can create ambiguity with respect to the true ownership of the data. It has been shown that for copyright protection applications, watermarks need to be noninvertible. The requirement of noninvertibility of the watermarking technology implies that it should not be possible to extract a watermark from a non-watermarked document.

- Copy attack: The goal of copy attack is not to destroy the watermark or impair its detection, but to estimate a watermark from watermarked data and copy it to some other data, called target data. The estimated watermark is adapted to the local features of the target data to satisfy imperceptibility. The copy attack is applicable when a valid watermark in the target data can be produced with neither algorithmic knowledge of the watermarking technology nor knowledge of the watermarking key.

Counter-Attacks

There are many ways to increase robustness against attacks. These include [5,43]:

- Image registration or synchronization: The received image data has to be mapped to the original host image in order to determine the locations where the watermark has to be extracted. Imperfect image registration can result from cropping and other geometric attacks or from synchronization problems. Image registration can be seen as a separate state prior to watermark detection. To survive against geometrical transformations, there have been much research on embedding specific signals or designing a different structure in the watermark.

- Inverting Distortions: This is to invert any processing that has been applied since the watermark was embedded. It includes rescaling and gamma correcting the amplitude of the image,

- Error correction coding [41]: Because images go through attacks, an error can occur in the message bits. We can increase decrease the error in the message by encoding the message using an error correcting code. By defining the mapping between messages and code words in an appropriate way, it is possible to build decoders that can identify the code word closest to a given corrupted sequence.

1.3 Properties of Watermarking Systems

For evaluating watermarking systems, they can be characterized by a number of defining properties [14, 32, 53]. The suitability of a given watermarking system for a given application may be judged in terms of the following properties.

1.3.1 Robustness

Robustness is the ability of the watermark to survive normal processing of images. The more robust a watermark, the more difficult it is to remove from the watermarked signal. We draw a distinction between robustness, which refers to common operations, and security, which refers to hostile operations. Many authors do not draw this distinction and use robustness to refer to both types of operations.

Like perceptual fidelity, actual robustness is difficult to quantify, in part because many attacks are difficult to model. Many watermarking papers report robustness by measuring bit or detection error rate of the embedded watermark after performing specific attacks. Attack benchmarks, such as StirMark [51, 52] and Checkmark [54], can be used to compare the robustness of image watermarking techniques after a common set of attacks have been performed.

1.3.2 Data Payload

Data Payload N_m is the number of bits a watermark encodes within a unit of time or within a media element. The more information one wants to embed, the lower is the watermark robustness. A watermark that encodes N bits is referred to as an N-bit watermark. Such a system can be used to embed any on of 2^N different messages. The output of the detector will have $2^N + 1$ output values: 2^N messages plus “no watermark present”.

Data payload is an important parameter since it directly influences the watermark robustness. The more information one wants to embed, the lower is the watermark robustness. The data payload is dependent on the application and does not include the extra bits for error correction codes.Copy control applications may require 4 to 8 bits of information to be received over 5 minutes while broadcast monitoring require 24 bits within the first second of commercials for video. It is suggested in that to avoid small scale proprietary solutions, roughly 70 bits of information should be embedded in an image [42].

1.3.3 Embedding Effectiveness

Embedding effectiveness is the probability that the embedder will successfully embed a watermark in a randomly selected media element. In other words, the effectiveness is the probability of detection immediately after embedding. This definition implies that a watermarking system might have an effectiveness of less than 100%.

Embedding effectiveness depends on the watermarking algorithm, data payload, image size, and image characteristics. For example, multiplicative spread spectrum algorithm [1] does not embed a watermark in a signal if the amplitude of that signal is constant. Also, if the image size is small, watermark cannot be embedded effectively. This is used in the mosaic attack [29].

1.3.4 Fidelity

Fidelity is the perceptual similarity between the original and watermarked versions of the media element. Visible Artifacts of an watermarked image may reduce or destroy the commercial value of the watermarked image. The most popular fidelity measure for images is the mean square error (MSE) [22]. MSE for images with M number of color components and N the number of pixels is [55],

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(s(i,j) - y(i,j)\right)^2$$

, where s is the original image and y is the modified image. More popular form of MSE is the peak signal to noise ratio (PSNR) [22]. For 8 bit images, it is given as follows:

$$PSNR = 10\log_{10}\frac{255^2}{MSE}$$

There is a tradeoff between the watermark robustness and fidelity. Increased robustness requires a stronger embedding, which in turn decreases the fidelity of the image. Because PSNR [22] does not correlate too well with the human visual system (HVS), other more sophisticated objective fidelity metrics have been used for watermark evaluation [2, 54, 56–58].

1.3.5 Cost

Cost is the computational cost of the embedder and detector. The economics of deploying watermark embedders and detectors can be extremely complicated and depends on the business models involved [59]. Among the many issues of concern are speed, memory usage of the embedder and detector. In the case of MPEG video broadcast, the watermark detector must be fast to allow real-time detection. For digital camera applications, the embedder must be fast and have a small memory footprint.

1.4 Performance Evaluation

For designing digital watermarking methods, an important and often neglected issue is proper evaluation and benchmarking [42,60]. This lack of a proper evaluation in designing watermarking methods causes confusion among researchers and hinders the adoption of digital watermarking in various applications. With a well-defined benchmark, researchers and watermarking software manufacturers would just need to provide a table of results, which would give a good and reliable summary of the performances of the proposed scheme for end users. To address this issue, a few still image digital watermarking benchmarks have been proposed. These include StirMark [51, 52, 60], Certimark [61], Checkmark [54] and Optimark [62]. These benchmarks evaluate robust watermark algorithms which are watermarks designed to survive normal signal processing applications which all also called attacks [5]. So these benchmarks include many attacks that are encountered in various applications. These benchmarks contributed significantly to the advancement of watermarking algorithms.

1.5 Contribution of this Dissertation

This dissertation contributes to the fair and systematic evaluation of robust still image watermarking in terms of fidelity and robustness. The main contribution of this thesis, as described in later chapters are:

- **Mean square error fidelity metric for presentation attacked images**

Evaluating fidelity of the attacked images is important for attack development and consequently watermark development. One element that watermark benchmarks still lacks is objective fidelity evaluation of the presentation attacked images. Presentation attacks include pixel value operations (e.g., histogram equalization, gamma correction, and contrast enhancement), geometrical attacks (e.g., rotation, scaling, random local bending), and pixel loss attacks (e.g., cropping, block loss in image transmission). For pixel value operations, the SSIM metric [63] and the Visual Quality metric [64] is insensitive to linear amplitude scaling and minor mean shifts but does not consider nonlinear operations. For geometrical attacks, it has been proposed to evaluate images through image registration techniques [57]. The complex wavelet SSIM metric [65] is also insensitive to minor geometrical transforms but is sensitive to rotation and scaling.

We describe a technique to measure fidelity of images in terms of mean square error in this dissertation by compensation techniques commonly used in the human visual system (HVS). We use conditional mean to evaluate images that went through pixel value operations. For pixel loss attacks, we use error concealment. For geometrical attacks, instead of using image registration techniques, we assume that we can approximate the mapping from the original image to the attacked image. This is true in a watermarking evaluation framework.

- **New summary statistics for still image watermark evaluation**

  Current watermarking benchmarks use summary statistics to summarize the results. One example is the average. Average bit error rate is used to summarize the bit error rates and average PSNR is used to summarize the PSNR values. This does not reflect still watermarking scenarios where each watermarked image is distributed to different people and distributions of the bit error rates and PSNR values become important.

Another approach to summarization is to use a threshold. We could measure the maximum bit error rate and compare it against a threshold. For fidelity, we could compare the fidelity results against the just noticeable difference (JND) [5, 6] value. But requiring the watermarking system to meet a hard threshold requirement puts too much restriction on the watermarking application.

In quality engineering [66, 67], it is recommended that we should also consider the variance of the distribution as well as the mean. We propose PSNR and BER summary statistic using the Taguchi loss function [66–68] which considers both the mean and variance for watermark evaluation. We show a watermark evaluation result using the new summary statistic.

- **Watermark Evaluation Testbed**

  While digital watermarking has received much attention within the academic community and private sector in recent years, it is still a relatively young technology. As such there are few widely accepted benchmarks that can be used to validate the performance claims asserted by members of the research community. This lack of a universally adopted benchmark has hindered research and created confusion within the general public. To facilitate the development of a universally adopted benchmark, we are developing at Purdue University a web-based system that will allow users to evaluate the performance of watermarking techniques. We refer to this system as the Watermark Evaluation Testbed or *WET* [58, 69]. This system consists of reference software that includes both watermark embedders and watermark detectors, attack scenarios, evaluation modules and a large image database. The ultimate goal of the current work is to develop a platform that one can use to test the performance of watermarking methods and obtain fair, reproducible comparisons of the results. We feel that this work will greatly stimulate new research in watermarking and data hiding by allowing one to demonstrate how new techniques are moving forward the state of the art.

2. WATERMARK FIDELITY EVALUATION

2.1 Introduction

One important property for watermarks is fidelity [5], which is defined as the perceptual similarity between the host image and the watermarked image. Although there are much research on how to embed watermarks imperceptibly, there have been limited research on evaluating image fidelity for attacked watermarked images [57]. Evaluating the fidelity of attacked images is important for developing new attacks and watermark algorithms. Two approaches to fidelity evaluation are subjective evaluation and using objective visual quality metrics [57].

The subjective method involves viewers who grade host, watermarked, and attacked image presented [57]. The viewer is asked to rate the fidelity of an image, using procedures such as the single/double-stimulus methods and the two alternatives forced choice. The ITU Recommendation 500 gives recommendations for standardized subjective image quality evaluation procedures. Since subjective quality assessment methods involve humans, they are not suitable for automatic benchmarking.

To measure fidelity for an automated watermark benchmark, we need to use objective measures. One of the widely used objective metric is the mean square error (MSE) and its variant PSNR (peak signal to noise ratio). It is well known that MSE (mean square error) fails to evaluate valumetric attacks such as gamma correction and amplitude scaling or geometrical attacks such as spatial shifts and the affine transform in terms of fidelity [57]. This is also true for other image quality metrics developed [26,63–65,70], although a moderate amount of geometric distortion is allowed in [65]. These metrics are developed to compare different signal processing operations such as blurring and image compression and are not designed to cope with presentation attacks described in subsubsection 1.2.3.

2.2 Conditional Entropy as a Fidelity Measure

If the watermarked image undergoes attacks such as amplitude scaling and gamma correction, mean square error cannot adapt to such effects. In image registration, mutual information has been successfully used to measure the alignment of images obtained through different modalities [71–73]. An advantage of mutual information [74] over mean square error is that it does not depend on the exact value of pixel values but only on the joint histogram of two images. For point operations, we can think of the original image and the attacked image as images obtained through different modalities. This motivated us to consider mutual information a fidelity measure to evaluate still image watermarking algorithms and attacks [58].

The entropy $H(X)$ of a discrete random variable X is defined by [75]

$$H(X) = -\sum_x p(x) \log p(x)$$

where $p(x)$ is the probability mass function of the random variable X. The joint entropy $H(X, Y)$ of a pair of discrete random variables (X, Y) with a joint distribution $p(x, y)$ is defined as

$$H(X,Y) = -\sum_x \sum_y p(x,y) \log p(x,y).$$

The conditional entropy $H(X|Y)$ is defined as

$$H(X|Y) = -\sum_x \sum_y p(x,y) \log p(x|y).$$

The properties of H include

$$\begin{aligned} H(X) &\geq 0 \\ H(X|Y) &\leq H(X) \\ H(X,Y) &= H(Y) + H(X|Y). \end{aligned}$$

The mutual information $I(X;Y)$ between two random variables X and Y is defined as

$$I(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

Properties of $I(X;Y)$ are

$$\begin{aligned} I(X;Y) &= \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= \sum_x \sum_y p(x,y) \log \frac{p(x|y)}{p(x)} \\ &= H(X) - H(X|Y) \\ &= H(X) + H(Y) - H(X,Y). \end{aligned}$$

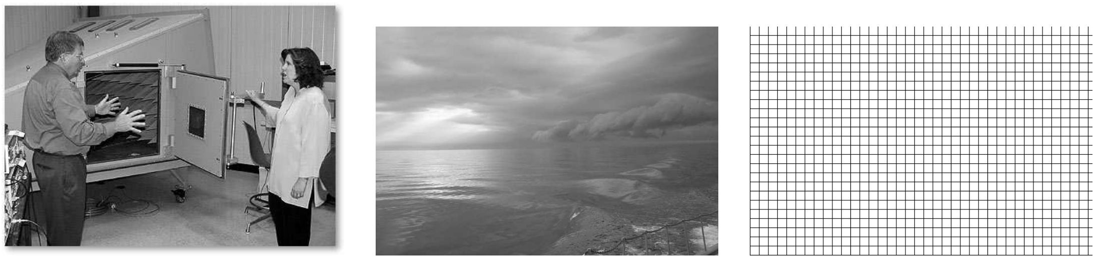
$I(S;Y)$ is a measure of the amount of information that Y contains about S [75]. $I(S;Y)$ is a "larger is better" measure because it is the reduction in the uncertainty of S by knowing the distribution of Y. To use mutual information in an environment where "smaller is better" such as the Taguchi loss function [66, 67, 69] described in chapter 3, we need to modify it. In this sense, we could use the conditional entropy $H(S|Y)$ as a similarity measure. $H(S|Y)$ is the average bits needed to describe S given Y. The relation between $H(S|Y)$ and $I(S;Y)$ is given as follows:

$$\begin{aligned} H(S|Y) &= H(Y,S) - H(Y) \\ &= H(S) - I(Y;S). \end{aligned}$$

We can estimate entropies by approximating the probabilities using histograms [72]. Let $h(s,y)$ be the joint histogram of the pixel values. We approximate the probability as follows:

$$\begin{aligned} p(x,y) &\approx \frac{h(x,y)}{\sum_x \sum_y h(x,y)} \\ p(x) &= \sum_y p(x,y) \\ p(y) &= \sum_x p(x,y). \end{aligned}$$

While there is a link between minimum mean square error and mutual information [76], it depends on the noise distribution of the attack channel and the



(a) fcc-door(440x324) (b) 0115(600x400) (c) grid-16-16(600x400)

Fig. 2.1. The test images used for comparing different compensation techniques. "fcc-door," "0115," and "grid-16-16."

distribution of the original image. In the next section, we describe a technique to measure fidelity using a conditional mean which also uses the joint histogram of two images.

2.3 Mean Square Error Fidelity Evaluation through Compensation

In the following, we will extend MSE by assuming the attacker has information the watermark detector/decoder may not have and will use it to restore the image to its original form. Specifically, we will modify MSE to compensate valumetric attacks, geometrical attacks, and pixel loss attacks. A distortion function we will use is given below and similar to the one given in [55].

$$d(\mathbf{s}, \mathbf{y}) = \min_{\theta \in \Theta} \|\mathbf{s} - T_\theta \mathbf{y}\|^2$$

, where Θ is the set of compensating functions. We develop T_θ for geometrical attacks, valumetric attacks and pixel loss attacks and combine them to measure in terms of MSE. For test images, we will use the image shown in Figure 2.1 as our test image.

2.3.1 Compensation for Valumetric Attacks

Valumetric Attacks or point operations are zero memory operations where we use a monotonically increasing or decreasing function $f(s)$ to map a pixel value s to $f(y)$ [22,77] for all pixels in an image. Valumetric attacks include contrast stretching, digital negative, range compression, and histogram equalization. We assume that these operations does not change the content in an image because human visual system (HVS) determines the content not by the exact value of a pixel but whether the pixel value is greater or smaller than the neighborhood pixel values. This is analogous to changing keys in music. We assume that the attacker knows $f(s)$ and will use that information to invert to the original image. Example of this is display of images on different computers. If you have an image on your computer that you want to display on another computer, you may want to apply color correction to that image to show a similar image on the other computer [78]. To apply color correction, you need to know the display characteristics of both computers. This information are stored in the form of color profiles and this information may not given to the watermark detector.

Let $\hat{s}(y)$ be a function that maps the attacked image pixel values to other pixel values. This function is defined only for values y such that $p(y) > 0$. We can write MSE for the image produced by the mapping $\hat{s}(y)$ using the joint probability as follows:

$$MSE(s,y) = \sum_{s} \sum_{y,p(y)>0} (s - \hat{s}(y))^2 p(s,y).$$

It is well known that $\hat{s}_{mmse}(y)$ ($p(y) > 0$), which minimizes MSE, is a conditional mean [79],

$$\hat{s}_{mmse}(y) = \sum_{s} sp(s|y).$$

Theorem 2.3.1 *Let $y = f(s)$ be a monotonically increasing function. Then $\hat{s}_{mmse}(y)$ is an increasing function for values of y s.t. $p(y) > 0$.*

Proof We can rewrite $\hat{s}_{mmse}(y)$ as follows:

$$\hat{s}_{mmse}(y) = \sum_{s,p(s)>0,f(s)=y} sp(s|y).$$

Because $\sum_s p(s|y) = 1$, $\hat{s}_{mmse}(y)$ is an weighted average of s's for $f(s) = y$. By the definition of f, if s_1 and s_2 exists s.t. $f(s_1) > f(s_2)$ then $s_1 > s_2$. This implies that if $y_1 > y_2$, then $\hat{s}_{mmse}(y_1) > \hat{s}_{mmse}(y_2)$ for any y_1 and y_2 s.t. $p(y_1) > 0$ and $p(y_2) > 0$. This means $\hat{s}_{mmse}(y)$ is an increasing function for values of y s.t. $p(y) > 0$. ■

Corollary 2.3.1 *Let $y = f(s)$ be a monotonically increasing function. If we define mutual information $I(s;y)$ as [75]:*

$$I(s;y) = \sum_s \sum_y p(s,y) \log \frac{p(s,y)}{p(s)p(y)},$$

then $I(s;y) = I(s;\hat{s}_{mmse}(y))$.

Proof Because $\hat{s}_{mmse}(y)$ is an increasing function of y, it is a one-to-one function of y s.t. $p(y) > 0$. If $\hat{s}_{mmse}(y)$ is a one-to-one function, then $p(s,y) = p(s,\hat{s}_{mmse}(y))$ and $p(y)$=$p(\hat{s}_{mmse}(y))$, for y s.t. $p(y) > 0$. By the definition of $I(s;y)$, $I(s;y) = I(s;\hat{s}_{mmse}(y))$. ■

Above theorem and corollary can also be proven for the monotonically decreasing case, similarly.

Since a conditional mean preserves the rank of the attacked image pixel values and has the minimum mean square among zero memory operations, we use it to compensate images that go through a point operation. From the histogram $h(s,x)$ of pixel values, we can approximate the conditional mean as follows:

$$\begin{aligned}
p(s,y) &\approx \frac{h(s,y)}{\sum_s \sum_y h(s,y)} \\
p(y) &= \sum_s p(s,y) \\
&\approx \frac{\sum_s h(s,y)}{\sum_s \sum_y h(s,y)} \\
p(s|y) &= \frac{p(s,y)}{p(y)}
\end{aligned}$$

$$\begin{aligned} &\approx \frac{h(s,y)}{\sum_s h(s,y)} \\ \hat{s}_{mmse}(y) &= \sum_s sp(s|y) \\ &\approx \frac{\sum_s sh(s,y)}{\sum_s h(s,y)}. \end{aligned}$$

2.3.2 Compensating for Pixel Loss Attacks

We define pixel loss attacks as attacks that lose the value of a pixel. Pixel loss attacks include cropping, random column and row removal. Pixel loss attacks can be seen as a packet loss due the error occurring in the channel. Although we could use conditional mean for the lost pixels by mapping the lost pixels to an arbitrary pixel value (e.g. -1 or 256), we are ignoring the correlation between adjacent pixels in an image. The human visual system can estimate the values of loss pixels by the pixels close to the lost pixel. Here, we assume that the attacker knows exactly where the pixels were lost and will use that information to recover those lost pixels. Example of this is video de-interlacing [80]. In de-interlacing, we make full frames out of fields by interpolating lost pixels using adjacent pixels. The attacker who makes interlaced video from non-interlaced video knows that the video is interlaced but the watermark detector does not have this information.

For pixel loss attacks, there are error concealment methods already developed [81]. Here we use the neighborhood mean for error concealment [82] as shown in Figure 2.2. Algorithm is implemented as follows:

1. Initialize binary map $n(i,j) = m(i,j) = 0$, if pixel at location (i,j) is lost for image $y(i,j)$, else $n(i,j) = m(i,j) = 1$.

2. For all (i,j), $ec(i,j) = y(i,j)$.

3. Define neighborhood mean of $ec(i,j)$ as

   $$NM(ec(i,j)) = \frac{\sum_{k=-1}^{1}\sum_{l=-1}^{1} ec(i+k,j+l)I(m(i+k,j+l) > 0)}{\sum_{k=-1}^{1}\sum_{l=-1}^{1} I(m(i+k,j+l) > 0)}$$

   , where I is the indicator function.

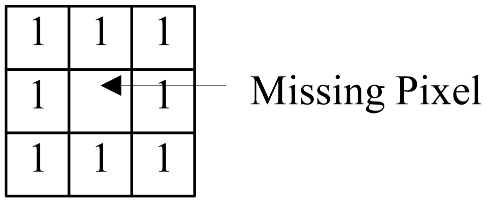


Fig. 2.2. The weights used for a neighborhood mean for error concealment.

4. For all (i,j), if $m(i,j) = 0$ and $\sum_{k=-1}^{1}\sum_{l=-1}^{1} I(m(i+k,j+l) > 0) > 0$, then $ec(i,j) = NM(ec(i,j))$ and $n(i,j) = 1$.
5. For all (i,j), m(i,j)=n(i,j).
6. Repeat 4 and 5 until $m(i,j) = 1$ for all (i,j). $ec(i,j)$ is the error concealment image.

2.3.3 Compensation for Geometrical Attacks

A geometric attack is defined by a spatial transformation [83]. It can be expressed as

$$[x, y] = [X(u, v)Y(u, v)]$$

or

$$[u, v] = [U(x, y)V(x, y)]$$

where [u,v] is the input image coordinates and [x,y] is the output image coordinates of the spatial transformation. X and Y are the forward mapping and U and V are the inverse mapping. Inverse mapping as shown in Figure 2.3 is more common than the forward mapping in spatial transformation implementations [83] and is used in StirMark 4.0.

Currently, two methods exist to evaluate geometrically attacked images [57]. One is to do a subjective evaluation. The other is to use a registration technique to match the host image and the attacked image geometrically. For image registration, the geometrical attacks are modeled as an affine transform locally and based on how large each affine transform covered each area determines the fidelity.

As mentioned above, we take the approach of compensating the attacked image and measure fidelity in terms of MSE. We assume that the attacker knows the geometric attack and will invert it. An example of this is different pixel aspect ratios used in broadcast and computers [78]. To display video from one pixel aspect ratio to another aspect ratio, we need to apply interpolation in the horizontal or vertical direction which is a form of geometrical attack. We assume that the attacker knows the pixel aspect ratio and will use it to display video with the correct pixel aspect ratio. The watermark detector may not have this information. For all geometric attacks in StirMark 4.0, we can obtain the exact expression for forward mapping except the local random bending attack. This eliminates the image registration step. The reason that there is no exact expression for the forward mapping in local random bending is that there are random components in the inverse mapping.

For interpolation of pixels, we use the biquadratic interpolation used in StirMark 4.0. Its speed and interpolation quality is between bilinear interpolation and bicubic interpolation [84]. Its interpolating kernel is given as follows:

$$h(s) = \begin{cases} -2|s|^2 + 1 & |s| < 1/2 \\ (|s|-1)(|s|-3/2) & 1/2 < |s| \leq 3/2 \\ 0 & \text{otherwise} \end{cases}$$

Forward Mapping for the Affine Transform

For an affine transform inverse mapping

$$\begin{aligned} u &= a_1 x + a_2 y + a_3 \\ v &= a_4 x + a_5 y + a_6 \end{aligned}$$

, the forward mapping is

$$x = \frac{a_5(u-a_3) - a_2(v-a_6)}{a_1 a_5 - a_2 a_4}$$

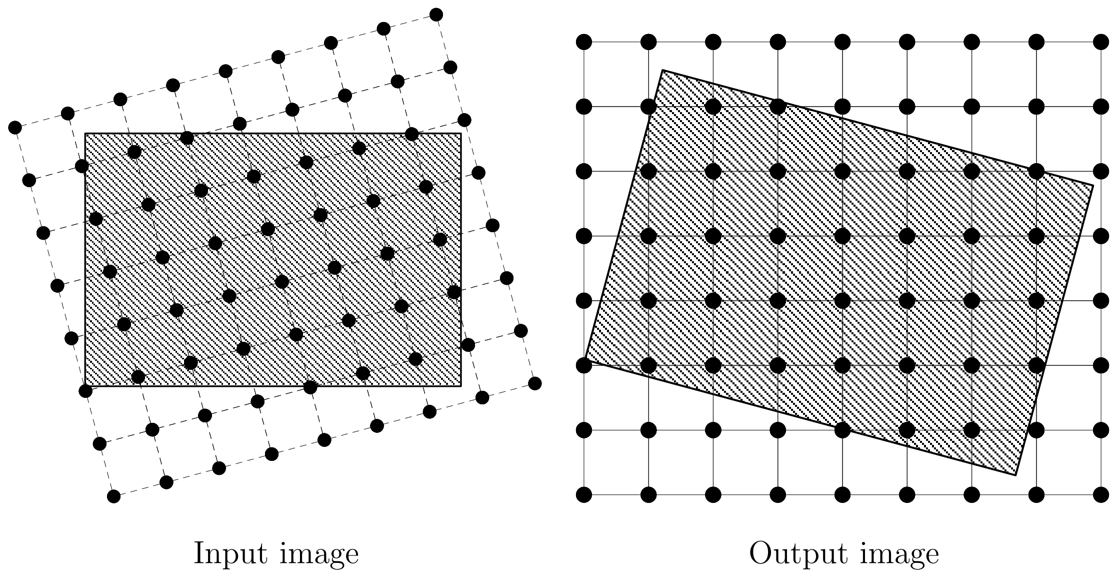


Fig. 2.3. An example of inverse mapping for implementing geometrical attacks.

$$y = \frac{-a_4(u - a_3) + a_1(v - a_6)}{a_1 a_5 - a_2 a_4}.$$

Inverse Mapping Approximation using the Bilinear Transform

To approximate the inverse mapping for the local bending attack, we interpolate the inverse mapping using the piecewise bilinear transform. Piecewise bilinear transform is assuming that the square grid in the output image coordinates has been inverse mapped from a quadrangle from the input image coordinates by a bilinear transform.

A bilinear transform is given as follows:

$$\begin{aligned} u &= a_1 x + a_2 y + a_3 xy + a_4 \\ v &= a_5 x + a_6 y + a_7 xy + a_8. \end{aligned}$$

A property of the bilinear transform is that it maps horizontal or vertical lines to a straight lines in the transformed coordinates [83]. This means that a grid square in the output image coordinates is mapped to a quadrangle in the input image coordinates.

Given four points on a square grid (x_0, y_0), $(x_0 + 1, y + 1)$, $(x_0 + 1, y_0 + 1)$, and its corresponding bilinear transformed points (u_1, v_1), (u_2, v_2), (u_3, v_3), and (u_4, v_4), we can obtain the bilinear transform as follows:

$$\begin{aligned} u &= a_1(x - x_0) + a_2(y - y_0) + a_3(x - x_0)(y - y_0) + u_1 \\ v &= a_5(x - x_0) + a_6(y - y_0) + a_7(x - x_0)(y - y_0) + v_1 \end{aligned}$$

$$\begin{aligned} a_1 &= u_2 - u_1 \\ a_5 &= v_2 - v_1 \\ a_2 &= u_3 - u_1 \\ a_6 &= v_3 - v_1 \end{aligned}$$

$$\begin{aligned} a_3 &= u_4 - a_1 - a_2 - u_1 \\ a_7 &= v_4 - a_5 - a_6 - v_1. \end{aligned}$$

Forward Mapping using the Inverse Mapping Approximation

There are two methods we know, on approximating the forward mapping given the piecewise bilinear transform. One method we did not implement is as follows: For each input image quadrangle inverse mapped by the output image grid square, we can determine what input image grid squares intersect with or are inside the input image quadrangle. Each input image grid square keeps a list of all the quadrangles it is inside or intersects with. Then, for a point (u, v) in the input image, we first find the input image grid square that includes (u, v). We go through the list of quadrangles that the input image grid square keeps and if we find a quadrangle that includes the point (u, v), we can obtain the forward mapping (x, y) using a closed form of the inverse bilinear transform [83].

The second approach, we implemented, does not use lists. Instead we use the variable metric algorithm [85] to find the inverse with the following cost function and its gradient:

$$f(x, y) = (U(x, y) - u)^2 + (V(x, y) - v)^2$$

$$\nabla f(x, y) = 2 \begin{bmatrix} (U(x, y) - u)(a_1 + a_3(y - y_0)) + (V(x, y) - v)(a_2 + a_3(x - x_0)) \\ (U(x, y) - u)(a_5 + a_7(y - y_0)) + (V(x, y) - v)(a_6 + a_7(x - x_0)). \end{bmatrix}$$

We used the Broyden-Fletcher-Goldfarb-Shanno (BFGS) variable metric algorithm from the gnu gsl library [86] as our variable metric algorithm. To use the variable metric algorithm, we need to choose an initial point (x_o, y_o). We make an array $\mathbf{a}_{ij}$ that stores all the initial points for the input grid points. $\mathbf{a}_{ij}$ has the same size as the input image. For each point (u, v) in the input image, we choose the initial point as $\mathbf{a}_{\lfloor u \rfloor \lfloor v \rfloor}$ and based on the value of the initial point we can determine f(x,y), and $\nabla f(x, y)$. For each output image grid square, we draw a bilinear transformed quadrangle on the vector array $\mathbf{a}_{ij}$. We use the same bilinear transform used in the inverse mapping. Quadrangles are drawn using the midpoint line algorithm [87]. The values of the quadrangle we draw with are the center point of the output image grid square which is a vector not a scalar value. To fill inside the quadrangle, we use the neighborhood mean shown in Figure 2.2. Using the neighborhood mean, we also fill $\mathbf{a}_{ij}$'s that are not inside any quadrangle generated by the output image grid squares.

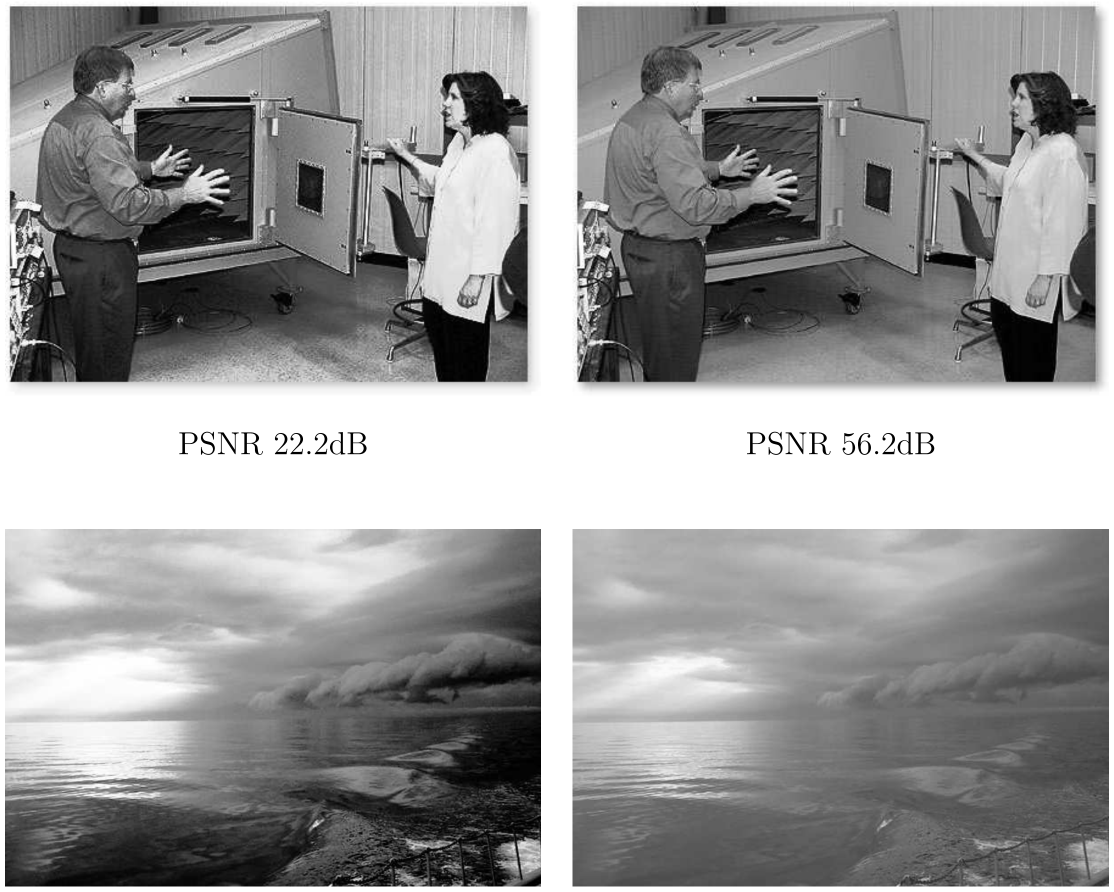
2.4 Experimental Results

For valumetric attacks, we chose histogram equalization and amplitude scaling and implemented the algorithm given in [22] for the histogram equalization. The histogram equalization images and their conditional mean images are shown in Figure 2.4. The amplitude scaling attack was implemented using the convolution filter given in StirMark 4.0. We applied the amplitude scaling to the sample test images and acquired the amplitude scaled images and their conditional mean image, as shown in Figure 2.5.

To show the effect of conditional mean compensation on PSNR for signal processing attacks, which are not valumetric, we chose gaussian filtering, sharpening, and JPEG compression. The implementation of these attacks are described in subsection 3.5.2. The Figure 2.6, Figure 2.7, and Figure 2.8 show the images and PSNR values for gaussian filtering, sharpening, and JPEG compression, respectively. The differences between the attacked image and conditional mean image are noted in the figures (c). We can see that conditional mean images show artifacts which are amplified in the difference image. This is due to the fact that our implementation of conditional mean does not consider the relationship between adjacent pixels or the frequency response of the human visual system. As we can see from the PSNR values, the conditional mean did not considerably change the PSNR values for attacks that are not valumetric attacks. This may be due to the fact that the three attacks locally preserve DC values.

To see the effect of conditional mean compensation and error concealment on pixel loss attacks, we chose cropping and row and column removal as our attacks. Figure 2.9 shows the images and their PSNR using conditional mean (b) and error concealment (c) for cropping attack with cropping factor 0.9 (a). Figure 2.10 shows the images and PSNR values using conditional mean (b) and error concealment (c) for column and row removal attack (a), where every 10th row and column are removed. It is shown that the PSNR values for the conditional mean images are similar for both attacks, which might be due to the effect of similar amount of pixels lost. Comparing cropping with row and column removal, row and column removal is shown to have better PSNR after compensation and this is because interpolation is usually more accurate than extrapolation. When comparing conditional mean and error concealment for cropping attack, results differ depending on the test image. PSNR values for the conditional mean image turned out to be greater than that of error concealment for the "fcc-door" test image. This might be due to the frame around the "fcc-door" test image. On the other hand, error concealment compensation is better for the "0115" test image, which might be due to the fact that the lost pixels have similar characteristics to the pixels near the boundary of the cropped image.

In order to investigate the effect of conditional mean compensation and error concealment on geometrical attacks, we selected zoom 25%, zoom 200%, 45 degree rotation and local random bending as our attacks. We used StirMark 4.0 to implement the attacks. The results for zoom 25%, zoom 200%, 45 degree rotation and local random bending are shown in Figure 2.11, Figure 2.12, Figure 2.13, and Figure 2.14. It is shown that if we compensate a zoom 25% image, it degrades the image considerably because we lose the high frequency components of an image. Figure 2.14 indicates that local random bending attack also includes pixel loss attacks due to cropping. Because of the many iterations in the forward mapping approximation, it



PSNR 22.2dB PSNR 56.2dB

PSNR 17.5dB PSNR 54.7dB

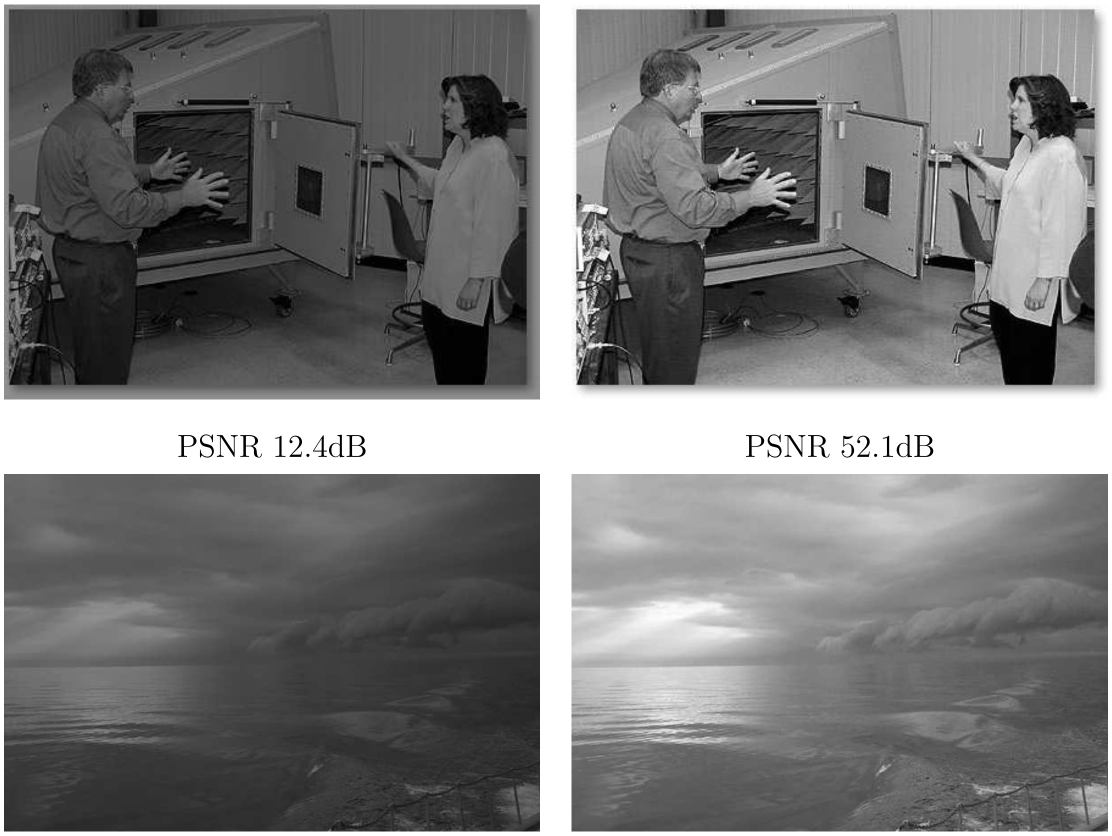
(a) Attacked image (b) Conditional mean

Fig. 2.4. Images and their PSNR values after histogram equalization attack: (a) is without compensation and (b) is with compensation using a conditional mean.

took about 30 seconds on a Xeon 2.4 GHz computer for a 600x400 size test image to compensate the local random bending attack.

2.5 Conclusion

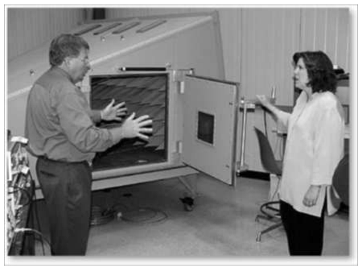
In this chapter, we described a technique to measure fidelity in terms of MSE for valumetric attacks including amplitude scaling and histogram equalization, pixel

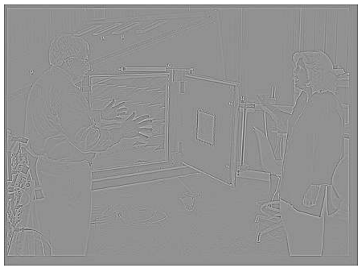


PSNR 12.3dB PSNR 51.8dB

(a) Attacked image (b) Conditional mean

Fig. 2.5. Images and their PSNR values after amplitude scaling attack with a factor of $\frac{9}{16}$: without compensation (a) and with compensation using a conditional mean (b).



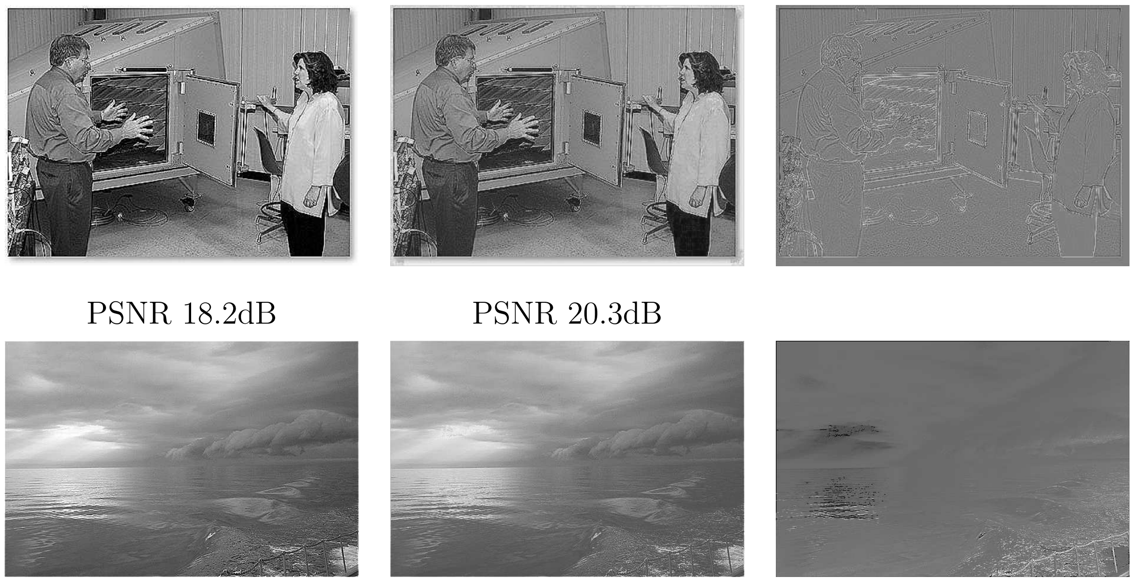


PSNR 28.3dB PSNR 28.6dB

PSNR 35.3dB PSNR 35.5dB

(a) Attacked image (b) Conditional mean (c) Difference

Fig. 2.6. Images and their PSNR values after gaussian filter attack: without compensation (a), compensation using a conditional mean (b), and the difference image (c) between (a) and (b).



PSNR 18.2dB PSNR 20.3dB

PSNR 24.8dB PSNR 27.0dB

(a) Attacked image (b) Conditional mean (c) Difference

Fig. 2.7. Images and their PSNR values after sharpening attack: without compensation (a), with compensation using a conditional mean (b), and the difference image (c) between (a) and (b).

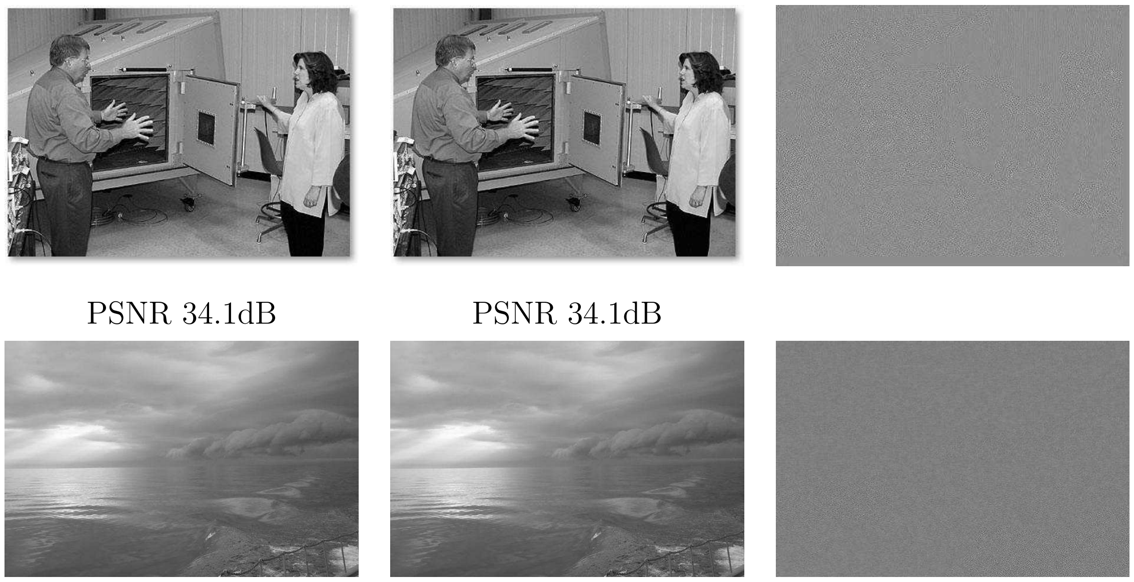
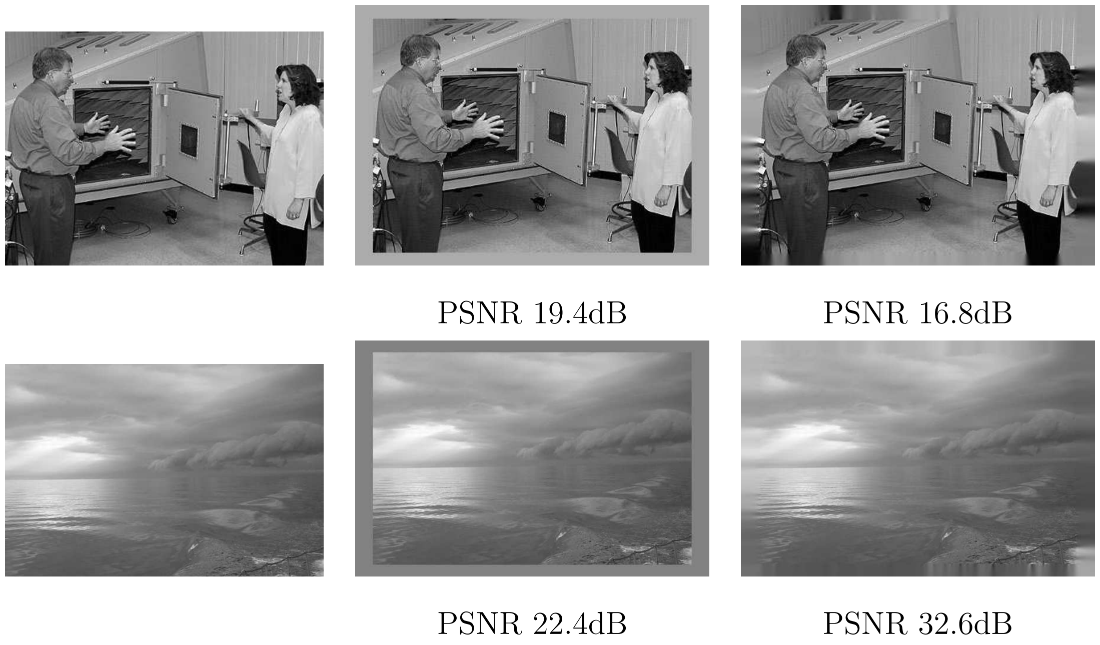
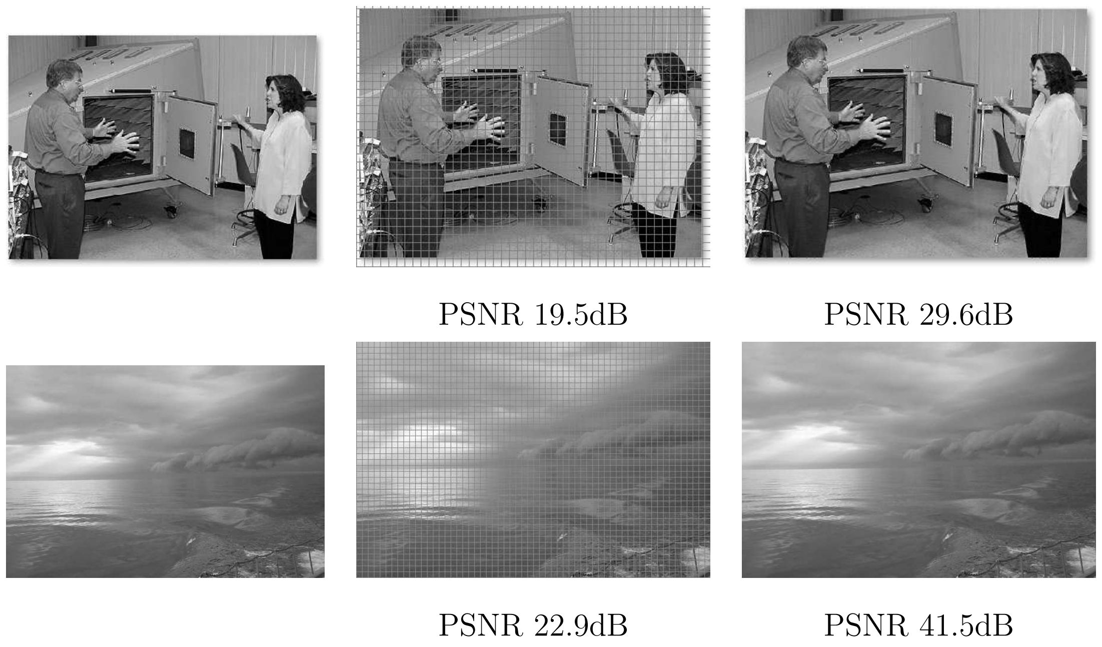


Fig. 2.8. Images and their PSNR values after JPEG Compression (Q=70): without compensation (a), with compensation using a conditional mean (b), and the difference image (c) between (a) and (b).



(a) Attacked image (b) Conditional mean (c) Error concealment

Fig. 2.9. Compensated images and their PSNR values for cropping attack with factor 0.9: cropped images (a), compensated by a conditional mean (b), and compensated by error concealment (c).

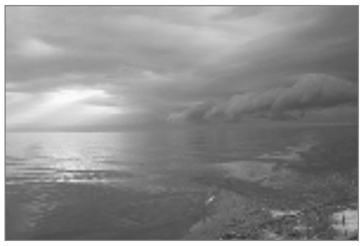


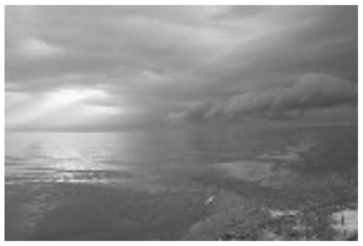
(a) Attacked image (b) Conditional mean (c) Error concealment

Fig. 2.10. Compensated images and their PSNR values for row and column removal attack: every 10th row and column removed (a), compensated by a conditional mean (b), and compensated by error concealment (c).

PSNR 21.6dB

PSNR 21.5dB



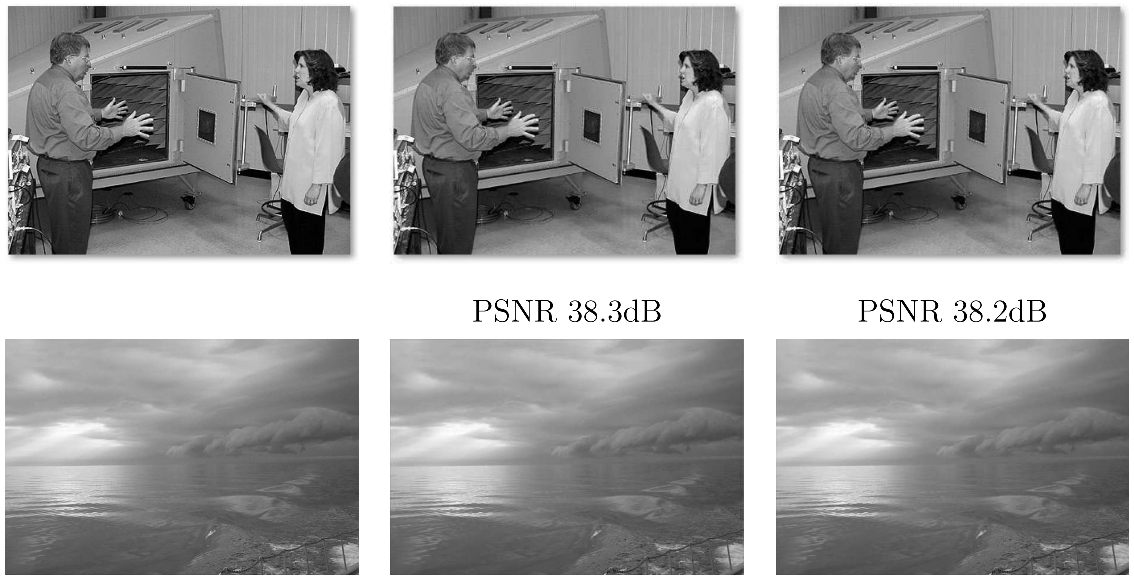


PSNR 29.6dB

PSNR 32.3dB

(a) Attacked image (b) Conditional mean (c) Error concealment

Fig. 2.11. Geometrically compensated images and their PSNR values for 25% zoom out attack: zoomed out images (a), pixel loss compensated by a conditional mean (b), and by error concealment (c).



(a) Attacked image (b) Conditional mean (c) Error concealment

Fig. 2.12. Geometrically compensated images and their PSNR values for 200% zoom in attack: zoomed in images (a), pixel loss compensated by a conditional mean (b), and by error concealment (c).

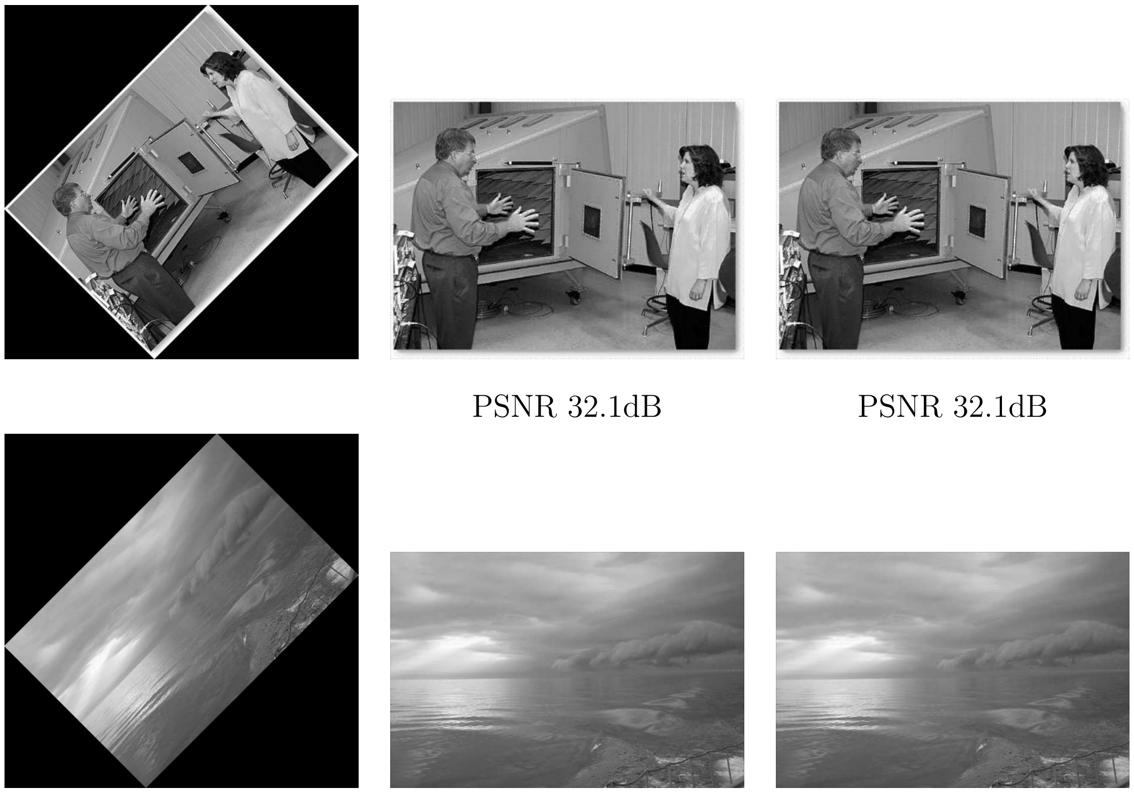
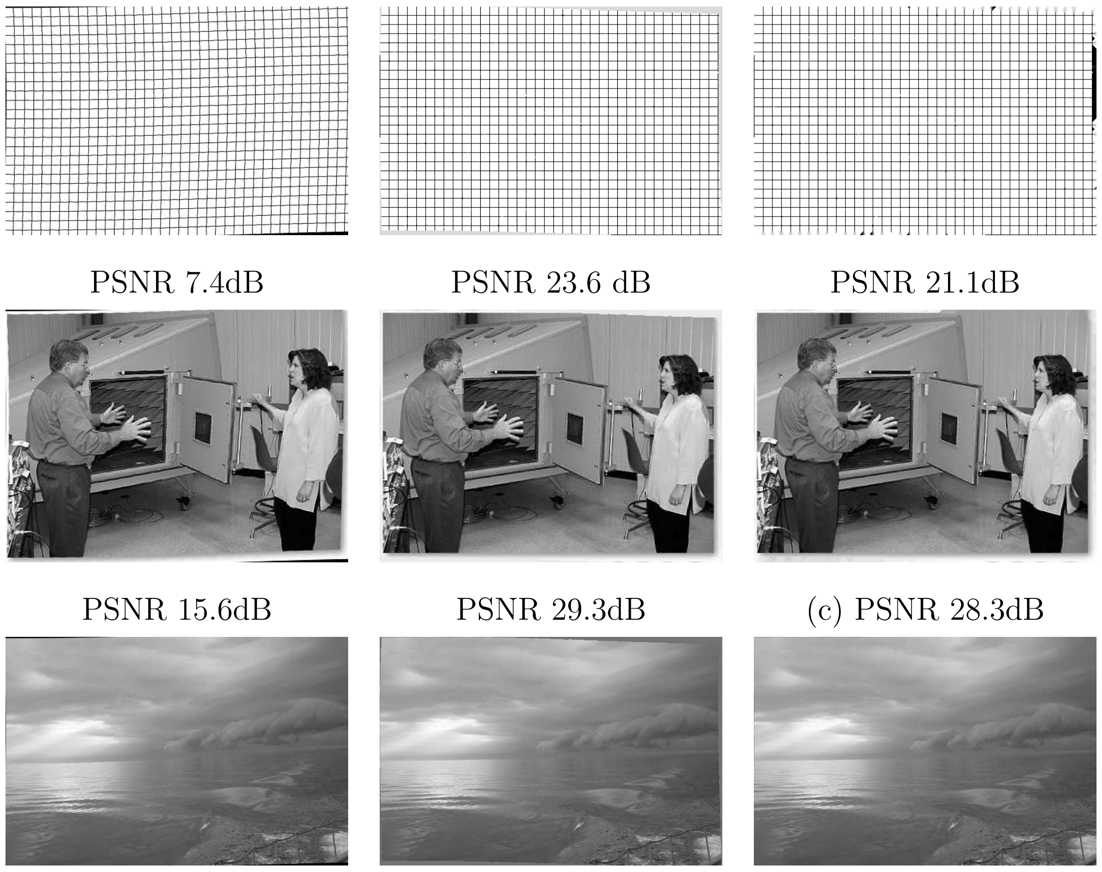


Fig. 2.13. Geometrically compensated images and their PSNR values for 45 degrees rotate attack: rotated images (a), pixel loss compensated by a conditional mean (b), and by error concealment (c).



PSNR 24.8dB PSNR 30.9dB PSNR 41.2dB

(a) Attacked image (b) Conditional mean (c) Error concealment

Fig. 2.14. Geometrically compensated images and their PSNR values for the local random bending attack: local random bending attacked images (a), pixel loss compensated by a conditional mean (b) and by error concealment (c).

loss attacks including cropping, and row and column removal, and geometrical attacks such as affine transform and local random bending. This technique uses the assumption that attacker knows the attack and will use that information to invert the attack. We gave some examples in previous sections where this assumption is true. To measure fidelity of valumetric attacked images in terms of MSE, we used conditional mean to compensate the attack. Using conditional mean, we can decrease the mean square error without affecting the pixel value order for valumetric attacks. We need to extend conditional mean to consider the frequency component of the image or correlation between adjacent pixels. For fidelity evaluation of pixel loss attacks, we assumed that the attacker knows where the pixels were lost and will use error concealment techniques to estimate the lost pixels. We compared error concealment with using only conditional mean. We only used a simple neighborhood mean to estimate the lost pixel values. Neighborhood mean seem to work better than conditional mean when the lost pixels have similar characteristics to the near preserved pixels. One suggestion to determine the MSE value of the pixel loss attacked image is to choose the minimum MSE for using error concealment and using conditional mean. To further reduce the MSE, we need to investigate different error concealment techniques such as multiresolution error concealment techniques. For geometrical attacks, we assumed that attacker knows the attack and will invert the geometrical attack. To invert the local random bending attack, we approximated the inverse mapping using the bilinear transform and found the forward mapping using a conjugate gradient method. To improve speed for forward mapping, we could use a bigger square size sacrificing the accuracy of the registration or use information from previously mapped pixels to better estimate the initial points.

3. WATERMARK EVALUATION

3.1 Introduction

As we have seen in Chapter 1, watermark designers design a watermark from various methodologies and algorithms [6]. While no technique exists which is better than others from all points of view, a comparative analysis of the techniques developed until now is very important. Such an analysis may be useful to show the advantages and disadvantages of different algorithms, to guide system designers in the choice of the most suitable technique for a given application, to guide research towards new more efficient schemes. This requires an watermark evaluation framework. Previous watermark evaluation frame work include StirMark [51, 52, 60], Checkmark [54] and Optimark [62].

StirMark [51, 52, 60] is the most popular reference benchmark for still image watermark evaluation. Images are watermarked with the strongest strength parameter α that is below a fidelity threshold. Threshold is chosen as PSNR 38 dB. Then a set of attacks is applied to the watermarked image, to look whether the watermark survives. Watermark detection and decoding is only treated successful when the whole message is recovered. The benchmark score is produced by assigning 1 to detection and 0 to non-detection. The average score is then computed for each class of attacks and the overall score is obtained by averaging partial scores. What StirMark did not consider was the false probability rate for using different keys and unwatermarked images.

Checkmark [54] includes many attacks that are not included in StirMark such as image denoising, wavelet compression, new geometrical attack, etc. Among 400 attacks in Checkmark, only 100 attacks are from StirMark. It also uses perceptual metric such as the Watson metric [26] or the weighted mask SNR(WMPSNR).

Optimark [62] considers the false positive error probability by using the receiver operating characteristic (ROC). It generates the ROC curve by assuming the positive detection probability and the negative detection error probability are Gaussian distributed. It also considers varying the message and keys for watermark evaluation.

For performance measures, receiver operating characteristic (ROC), bit error rate (BER), and message error rate are generally used in watermark evaluation [5,8,56]. Bit Error Rate (BER) is defined as the ratio of the wrongly extracted bits by the watermark decoder to the total number of embedded bits. To display evaluation results, the use of "BER versus visual quality," "BER versus attack," "attack versus visual quality" for a fixed BER, was proposed in [56]. In watermark evaluation, it is important to summarize the results to facilitate the comparison of algorithms. For BER, the results are summarized using message error rate or average bit error rate [56]. Message error rate is defined as the number of correctly decoded messages to the total number of watermarked image tested. For fidelity metric summarization, some iterate the embedding strength for each image to meet a fidelity requirement threshold [2,56,57,62]. To summarize ROC, area under the curve (AUC), equal error rate, false negative rate for a fixed false positive probability are used [56,62].

In 3.4, we will define and describe the watermark evaluation procedure we will use in this dissertation. We will evaluate watermark performance in terms of BER and Receiver Operating Characteristics (ROC). We will measure fidelity by PSNR. ROC analysis is described in section 3.2. We will also define new summary statistics for BER and $PSNR$ using the Taguchi loss function [67] in subsection 3.4.1.

3.2 ROC Analysis

Given any image, a watermark detector has to decide if the given image is watermarked [56]. This can be seen as a binary hypothesis testing in that the detector has to decide between the *alternative hypothesis* (the image is watermarked) and the *null hypothesis* (the image is not watermarked). In binary hypothesis testing, two kinds of errors can occur: accepting the alternative hypothesis, when the null hypothesis is correct and accepting the null hypothesis when the alternative hypothesis is true. The first error is often called *false positive* and the second error is called *false negative*. False positive probability is the probability that the watermark detector will detect a watermark in an unwatermarked image. We estimate the false positive probability by the false positive rate. The false positive rate FPR is defined as

$$FPR = \frac{FP}{TN + FP}$$

, where FP is the number of false positives, and TN is the number of true negatives. False negative probability is the probability that the detector will detect a watermark in an watermarked image. We estimate the false negative probability by the false negative rate. The false negative rate FNR is defined as

$$FNR = \frac{FN}{TP + FN}$$

, where FN is the number of false negatives, and TP is the number of true positives. We also define true negative rate $TNR = 1 - FPR$, and true positive rate $TPR = 1 - FNR$.

ROC graphs is used in signal detection to show the tradeoff between true positive rates and false alarm rates [88]. Usually in hypothesis testing, a test statistic is compared against a threshold to decide for one or the other hypothesis. ROC graphs are generated by repeating the test using varying decision thresholds. The ROC graph shows the relation between the *true positive rate (TPR)* on the y-axis and the *false positive rate (FPR)* on the x-axis. It is proposed in [56] that the same number of watermarked and non-watermarked should be tested.

An ROC curve is a two-dimensional description of detector performance. To compare different watermark detectors, we may want to summarize the ROC to a single value representing expected performance. As suggested in [56], we can use the integral under the curve to summarize the ROC. This is called the area under the curve (AUC) [88]. Since the AUC of an optimal detector will be 1.0, the value of a real detector will always be between 0 and 1.0. The AUC has an important statistical property [88] that it is an estimate of the probability that the detector will rank a randomly chosen watermarked image higher than a randomly chosen unwatermarked image. It is possible for a high-AUC detector to perform worse in a specific region than a low-AUC detector. But in practice, the AUC performs very well and is often used when a general measure of predictiveness is desired [88]. It is also useful to compare different watermarking detectors when the desired false alarm rate is not given. To obtain AUC, we use the algorithm described in [88].

3.3 Reliability Testing

Most of the watermark evaluation methods [2,54,56–58] fit the reliability testing framework. We define reliability as the probability that a watermarking algorithm will correctly detect or decode a watermark for a specified fidelity requirement under a given set of attacks and images. In reliability testing, a system is evaluated in terms of quality, load, capacity and performance [67]. We define quality as the fidelity of watermarked images produced by an watermarking algorithm and attacks. Measuring fidelity by the compensated mean square error (MSE) was described in chapter 2. We define capacity as the maximum data payload or minimum embedding strength that satisfies a certain error criteria. Then, we define load to be the actual embedding strength and data payload of a watermark. Because, capacity usually exceeds watermarking requirements, we will not consider capacity here. We described various performance measures used in watermark evaluation at the introduction of this chapter. In a reliability testing framework, we vary the load and measure quality and performance for different attacks and images and see if the performance meets the required criteria.

3.4 Watermark Evaluation Procedure

Watermark users usually fix the embedding strength α and threshold T, where the fixed values depend on the application of the watermarking system. The embedding Strength α depends on the fidelity requirement and the threshold T depends on the detection probability requirement of the application.

The input of a benchmarking system are the watermark embedding and the detection-decoding software and the output are the robustness characteristics. When evaluating watermarking algorithms, we fix the embedding strength and data payload for each algorithm and vary the images, keys and messages. This is the approach used by StirMark [51, 52, 89] and Optimark [62, 90]. Although, there is an option to iterate the embedding strength α such that the PSNR of each watermarked image is within +/- 1 PSNR of the target PSNR in Optimark, this will not be true for all watermarking algorithms because of processing time requirements.

We adopted a watermark evaluation procedure similar to Optimark [62, 90]. The benchmarking system's parameters are [62, 90]:

- a set of images $\mathbf{I} = \{I_i | i = 1 \dots N_I\}$
- a set of attacks $\mathbf{A} = \{A_j | j = 1 \dots N_A\}$
- N_K: Number of keys we use for each image
- a set of fidelity specifications $\mathbf{Q} = \{Q_l | l = 1 \dots N_Q\}$

From this, a set of $N_I \times N_K$ watermarked images for an attack A_j and fidelity specification Q_l, where fidelity specification is described below. For the message, we use the bits generated by a pseudo binary random number generator with the current time as the seed. To test as many keys as possible, we arbitrary use keys $k = 1, \dots N_I \times N_K$ to test the watermarking algorithm and use keys $(i-1)N_K+1, \dots, iN_K$ for each test image I_i, where $i = 1, \dots, N_I$. We will use PSNR as an image fidelity measure and Bit Error Rate (BER) and ROC curve with and without attacks as a measure of robustness in this paper. Let's define $PSNR_k, k = 1, \dots, N_K N_I$ to be the

PSNR of the kth watermarked image with no attack. For false positive probability, we measure the detection statistics of unwatermarked images.

3.4.1 BER and PSNR Summarization

Fidelity specification Q_l for an application is usually defined in terms of the specification limit. For example, we could define average PSNR as

$$\text{Average}(PSNR) = \frac{1}{N_K N_I} \sum_k PSNR_k$$

and require the average PSNR be greater than LSL=45dB, where LSL is the lower specification limit. Or we could define percentile as

$$\text{Percentile}(PSNR, \text{LSL}) = \frac{1}{N_K N_I} \sum_k I(PSNR_k \geq \text{LSL})$$

where I is the indicator function and then require Percentile$(PSNR, 40\text{dB}) > 99\%$, where 99% is the yield requirement. Summarizing PSNR of watermarked images using the average PSNR or percentile has its disadvantages. Figure 3.1 is an example that shows the disadvantage of specifying the fidelity requirements in terms of average PSNR. If we require Average$(PSNR) >$ LSL for an watermarking algorithm, both algorithm A and B satisfy the fidelity specification but algorithm B produces more watermarked images with relatively low PSNR values near the left tail of the distribution. Figure 3.2 is an example that shows the disadvantage of specifying the fidelity requirements in terms of percentile. If we require Percentile$(PSNR, \text{LSL}) = 1$, both algorithms do not produce watermarked images that have a PSNR value lower than LSL but algorithm A produces more watermarked images with relatively low PSNR images near LSL. A compromise between average and percentile is to specify the fidelity requirements in terms of Taguchi loss function.

For fair benchmarking, one has to ensure that the methods under investigation are tested under comparable conditions [56]. The shortcomings of using only the mean or percentile to define product requirements prompted Taguchi to formulate a continuous loss function [67] that more closely represents the quality degradation associated with the increased deviation of x, one of the many parameters used for product specification, from the optimal parameter value τ. The advantage of the Taguchi loss function over average and percentile is that it considers both the mean and variance of the distribution. The Taguchi method was used in [34] to build better watermarking algorithms and in [35] to select optimal parameters for video compression. Since PSNR and BER is defined for each watermarked image, we can define the fidelity and robustness requirements using the Taguchi loss function.

For a parameter x, and the optimal parameter value τ, the loss function is defined as [67]:

$$L(x) = k(x - \tau)^2$$

, where the coefficient k is determined by each application. For a random variable X with mean μ and variance σ^2, we obtain

$$E[L(X)] = k(\sigma^2 + (\mu - \tau)^2).$$

For many situations where a parameter value should be minimized, only an Upper Specification Limit (USL) is set. For these situations, Taguchi defines the smaller-is-better loss function as

$$L(x) = kx^2,$$

, with k determined by equating the loss function to the quality loss at the upper specification limit.

For performance characteristics where larger-is-better, only the Lower Specification Limit (LSL) is designated. The Taguchi loss function is then

$$L(x) = kx^{-2},$$

with k determined by equating the loss function to the quality loss at the lower specification limit.

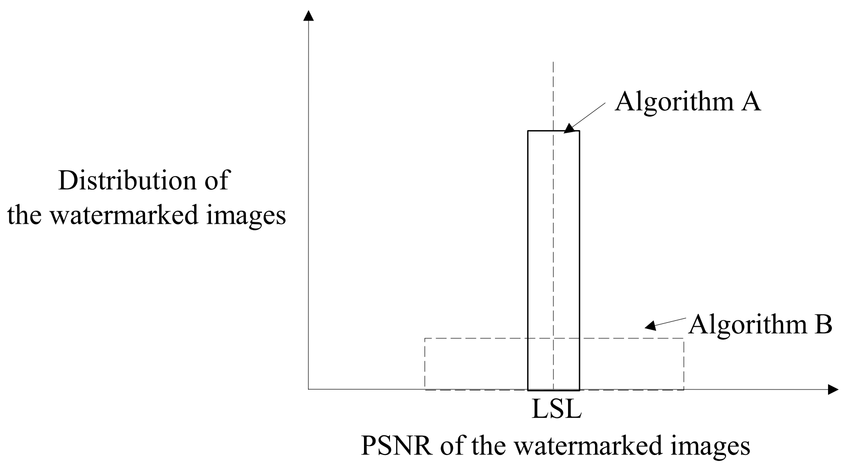

Algorithm A
Distribution of the watermarked images
Algorithm B
LSL
PSNR of the watermarked images

Fig. 3.1. An example showing the shortcomings of using average PSNR for a fidelity specification.

Since PSNR is a larger-is-better measure and BER is a smaller-is-better measure, we define a pseudo PSNR and a pseudo BER as follows using the Taguchi loss function as follows:

$$PSNR' = (\frac{1}{N_I N_K}\sum_k (\frac{1}{PSNR_k})^2)^{-\frac{1}{2}}.$$

$$BER^{'} = (\frac{1}{N_I N_K}\sum_k (BER_k)^2)^{\frac{1}{2}}.$$

We choose $PSNR'$ as fidelity summary statistic and BER' and AUC of the ROC curve as a robustness summary statistic for a watermarking technique for a set of images , **I**, keys=1, . . . , $N_I N_K$ for a given attack A_j and fidelity specification Q_l.

3.5 Example Evaluation Setting

For a base test setting, we set the lower specification limit for $PSNR'$ as 45dB. We chose the data payload N_d to be 16 bits. This specification can be used as a specification for "fingerprinting" applications [91]. We tested N_K=20 for each image for no attacks and false positives. We test N_K=2 keys for each image for

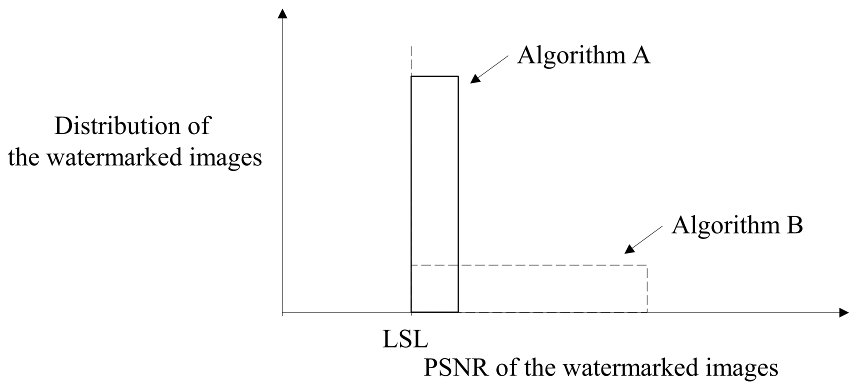

Algorithm A
Distribution of the watermarked images
Algorithm B
LSL
PSNR of the watermarked images

Fig. 3.2. An example showing the shortcomings of using percentile for a fidelity specification.

other attacks. For the image test set I, we use WET image database described in subsection 4.2.4. It has $N_I = 1301$ images.

3.5.1 Watermarking Algorithms

Let m_n=0, 1, ($n = 1, \ldots, N_m$) be the bits we want to embed. $\mathbf{X}_n$ is the watermarked vector, $\mathbf{S}_n$ be the original host vector or image, $\mathbf{W}_n$ is a normalized watermark vector ($\| \mathbf{W}_n \| = 1$). We define α as the scaling factor. For α, we use a fixed value for each test that satisfies the fidelity requirement. We find α using the bisection method [85]. In this experiment, we generate a normalized watermark vector by normalizing an i.i.d. Gaussian Random Vector with distribution $N(0, 1)$. We define $(\mathbf{X}_n)_i$ be the ith element of $\mathbf{X}_n$.

We embed multiple bits using Spatial Division Multiplexing. We divide the image into 8x8 blocks and apply the Discrete Cosine Transform (DCT) to each block [92]. We evenly and randomly distribute each blocks to form N_m groups. We construct an original host vector $\mathbf{S_n}$ by row concatenating all the blocks in the nth group excluding some frequency components. A frequency component selection scheme in DCT domain is used for embedding to reduce watermark visibility as shown in Figure 3.3. This is to reduce the visibility of the watermark in an image [6].

As test algorithms, we choose three watermarking algorithms. Each algorithm is described below. Each uses a Gaussian Watermark Pattern. Each algorithm is a blind watermarking algorithm, that is it does not require the original host image to detect the watermark. Summary of each algorithm in terms of embedding, decoding and detection is shown in Table 3.1, Table 3.2 and Table 3.3. As shown in Table 3.2, the three algorithms use the same decoding scheme.

Additive Spread Spectrum Watermarking (ASSW)

In many watermarking schemes, Additive Spread Spectrum Watermarking (ASSW) is the modulation technique used to embed the watermark [93] because of fidelity

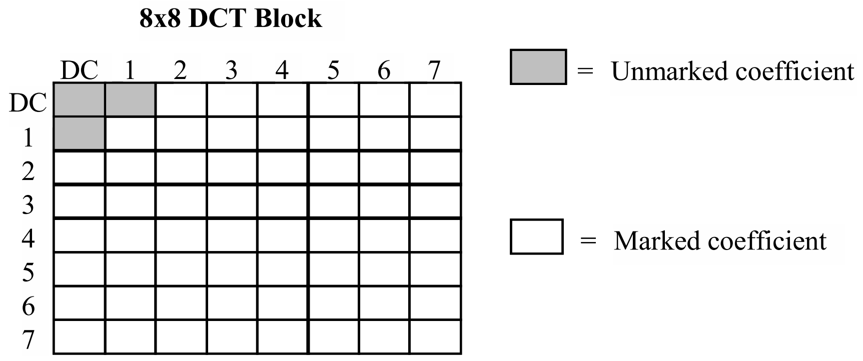


Fig. 3.3. A frequency component selection scheme in DCT domain for embedding to reduce watermark visibility.

Table 3.1
Watermark embedding for the three (ASSW, MSSW, ISSW) test algorithms.

Name	Embedding
ASSW	$\mathbf{X}_n = \mathbf{S}_n + \alpha(2m_n - 1)\mathbf{W}_n$
MSSW	$(\mathbf{X}_n)_i = (\mathbf{S}_n)_i + \alpha(2m_n - 1)\lvert(\mathbf{S}_n)_i\rvert(\mathbf{W}_n)_i$
ISSW	$\mathbf{X}_n = \mathbf{S}_n + (\alpha(2m_n - 1) - \mathbf{S}_n \cdot \mathbf{W}_n)\mathbf{W}_n$

requirements. In the simplest scheme, the bits composing the desired message are modulated by an spread spectrum sequence and added to the signal. Additive Spread Spectrum used here is a type of blind embedder [29]. The blind embedder is an embedder that ignores the host signal. In more elaborate schemes, differences in the host signal may be explored in order to reduce subjective distortion introduced by the watermark.

Multiplicative Spread Spectrum Watermarking (MSSW)

Multiplicative Spread Spectrum Watermarking of [94] is based on the non-blind multiplicative spread spectrum algorithm [1]. This method is based on the assumption that a watermark must be placed in perceptually significant components of a signal if it is to be robust to common signal distortions and malicious attacks.

Improved Spread Spectrum Watermarking (ISSW)

Improved Spread Spectrum Watermarking (ISSW) [93] removes the host signal interference in ASSW, producing a dramatic improvement in the quality of the watermarking process. The gains for the ISSW are similar to those obtained by Quantization Index Modulation (QIM) [38] for Gaussian noise attacks, but the method does not suffer from amplitude scaling which QIM is very sensitive.

Table 3.2
Watermark decoding for the three (ASSW, MSSW, ISSW) test algorithms.

Name	Decoding
ASSW	$\hat{m}_n = \begin{cases} 1 & \text{if } \mathbf{Y}_n \cdot \mathbf{W}_n > 0 \\ 0 & \text{otherwise} \end{cases}$
MSSW	$\hat{m}_n = \begin{cases} 1 & \text{if } \mathbf{Y}_n \cdot \mathbf{W}_n > 0 \\ 0 & \text{otherwise} \end{cases}$
ISSW	$\hat{m}_n = \begin{cases} 1 & \text{if } \mathbf{Y}_n \cdot \mathbf{W}_n > 0 \\ 0 & \text{otherwise} \end{cases}$

Table 3.3
Watermark detection for the three (ASSW, MSSW, ISSW) test algorithms.

Name	Detection	
ASSW	Watermark Present,	if $\sum\limits_n (2\hat{m}_n - 1)\mathbf{Y_n} \cdot \mathbf{W_n} > \mathrm{T}$
	Watermark Not Present,	otherwise
MSSW	Watermark Present,	if $\frac{\sum\limits_n (2\hat{m}_n - 1)\mathbf{Y_n} \cdot \mathbf{W_n}}{\sum_n \sum_i \lvert(\mathbf{Y_n})_i\rvert} > \mathrm{T}$
	Watermark Not Present,	otherwise
ISSW	Watermark Present,	if $\sum\limits_n (2\hat{m}_n - 1)\mathbf{Y_n} \cdot \mathbf{W_n} > \mathrm{T}$
	Watermark Not Present,	otherwise

3.5.2 Attacks

We use StirMark [51,52,60] 4.0 software for our attacks except histogram equalization which is an implementation of the described in [22]. We evaluated the algorithms for the following attacks:

- Gaussian filtering (blur): $\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} /16$
- Simple sharpening: $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$
- JPEG [25] compression with quality factor 70
- Amplitude scaling with scaling factor $\frac{9}{16}$
- Histogram equalization

We also measured performance for quality factor 50, 30, 10 to see the effect quality factor has on performance for JPEG attacks. StirMark 4.0 uses the software from the independent JPEG group [95].

3.6 Experimental Results

We define $PNSR_{cm}$ as the $PSNR$ obtained from the conditional mean image described in subsection 2.3.1. Figure 3.4 shows the 1/PSNR distribution of watermarked images for the 3 algorithms. Figure 3.5 shows the watermarked images for $PSNR' = 35dB$. The original image is shown in Figure 2.1. Table 3.4 shows the average PSNR, harmonic mean PSNR, $PSNR'$ values. We define harmonic mean PSNR as:

$$\text{Harmonic}(PSNR) = (\frac{1}{N_I N_K} \sum_k (\frac{1}{PSNR_k}))^{-1}.$$

Table 3.5 shows the results for the conditional mean PSNR. For the histogram equalization and amplitude scaling, it give a higher value than PSNR. Table 3.6 shows the bit error rates for different attacks. It shows that ISSW is always better than the ASSW in terms of bit error rate for the attacks selected except the histogram equalization. MSSW is better than ISSW and ASSW for the JPEG attack. It also shows that the sharpening attack lowered the bit error rate for all algorithms even though the images are degraded more than other attacks. Table 3.7 shows the 1-AUC results. It shows that 1-AUC values for ASSW and ISSW are similar and MSSW is better that the other two for JPEG and Gaussian filtering. Figure 3.6 shows the ROC results for false positives using unwatermarked images and false positives using different keys. In our test algorithms, it shows similar results. As shown in Figure 3.7, MSSW seems to be better than other algorithms in terms of false probability and false negative probability for attacks that preserve the low frequency components. This is due to the fact that the test images we use has energy concentrated in the low frequency components and MSSW embeds the watermark in the significant components of the images. Figure 3.8 shows the performance results for different payload and JPEG attack. As expected, the performance decreased when payload increased. Figure 3.9 shows the results for different JPEG attacks. It shows that performance of MSSW does not change as much compared to other two algorithms. Figure 3.10 shows the results for various embedding strength. It shows that ASSW and ISSW can have better performance than MSSW by sacrificing fidelity.

3.7 Conclusion

Robust image watermarks are image watermarks designed to survive attacks that include signal processing and spatial transformations. Most of the robust watermark evaluation methods evaluate watermarks in terms of load (payload, embedding strength), capacity (maximum payload, minimum embedding strength), quality (fidelity of the image), and performance. We evaluated three watermark algorithms by

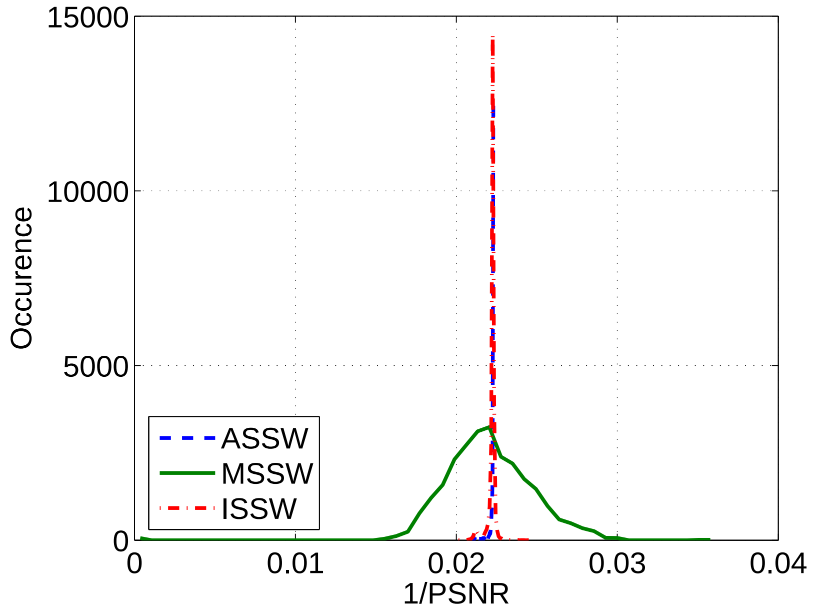
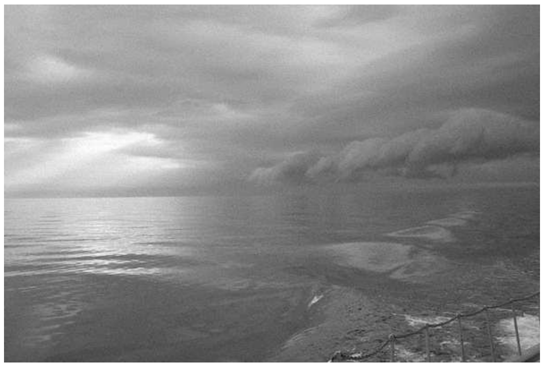


Fig. 3.4. The histogram (50 bins) of 1/PSNR values for the three test agorithms (ASSW, MSSW, ISSW).

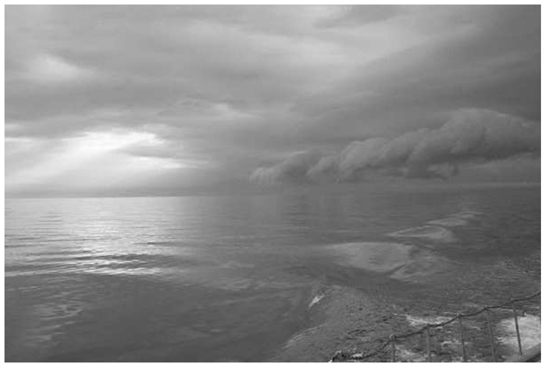
Table 3.4
The summary of PSNR values ($PSNR'$, Harmonic mean, Average) of three algorithms (ASSW, MSSW, ISSW) for different attacks.

Attacks	No Attack	Gaussian Filtering	Sharpening
ASSW	(45.0,45.0,45.0)	(31.6,32.4,33.5)	(21.9,22.2,22.8)
MSSW	(45.0,45.4,∞)	(31.6,32.3,33.5)	(22.5,22.9,23.8)
ISSW	(45.0,45.0,45.0)	(31.6,32.4,33.5)	(21.9,22.2,22.8)
Attacks	**JPEG**	**Amplitude Scaling**	**Histogram Equalization**
ASSW	(39.0,39.3,39.8)	(12.4,12.7,13.3)	(13.3,14.2,16.1)
MSSW	(38.1,38.4,38.9)	(12.4,12.7,13.3)	(13.0,14.3,16.3)
ISSW	(39.0,39.3,39.8)	(12.4,12.7,13.3)	(13.3,14.2,16.1)

ASSW(PSNR=34.8dB)



ISSW(PSNR=34.8dB)



MSSW(PSNR=41.8dB)

Fig. 3.5. Watermarked images and PSNR values of the "0115" test image using three test algorithms (ASSW, ISSW, MSSW).

Table 3.5
The summary of PSNR$_{\mathrm{cm}}$ values ($PSNR'_{cm}$, Harmonic mean, Average) of three algorithms (ASSW, MSSW, ISSW) for different attacks.

Attacks	No Attack	Gaussian Filtering	Sharpening
ASSW	(45.5,45.6,∞)	(32.6,33.2,34.1)	(24.1,24.3,24.7)
MSSW	(45.3,45.7,∞)	(32.5,33.1,34.0)	(24.7,25.0,25.6)
ISSW	(45.5,45.6,∞)	(32.6,33.2,34.1)	(24.1,24.3,24.7)
Attacks	JPEG	Amplitude Scaling	Histogram Equalization
ASSW	(39.3,39.6,∞)	(44.8,44.9,∞)	(44.5,44.5,44.5)
MSSW	(38.5,38.8,∞)	(44.3,44.5,∞)	(44.3,44.5,∞)
ISSW	(39.3,39.6,∞)	(44.8,44.9,∞)	(44.5,44.5,44.5)

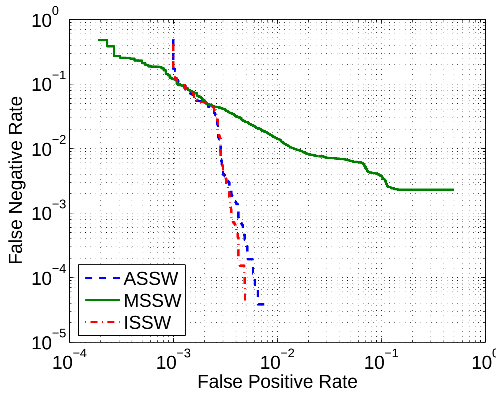
Table 3.6
The summary of BER values (BER', Average) of three algorithms (ASSW, MSSW, ISSW) for different attacks.

Attacks	No Attack	Gaussian Filtering	Sharpening
ASSW	(1.8e-2,1.4e-3)	(4.9e-2,1.1-e2)	(1.3e-2,4.8e-4)
MSSW	(3.3e-2,4.3e-3)	(9.6e-2,4.2e-2)	(3.3e-2,4.4e-3)
ISSW	(8.1e-3,2.8e-4)	(3.9e-2,6.7e-3)	(1.3e-2,4.6e-4)
Attacks	JPEG	Amplitude Scaling	Histogram Equalization
ASSW	(7.6e-2,2.4e-2)	(2.2e-2,2.0e-3)	(1.2e-3,2.4e-5)
MSSW	(5.1e-2,1.2e-2)	(3.3e-2,4.3e-3)	(3.2e-2,3.3e-3)
ISSW	(6.4e-2,1.7e-2)	(1.1e-2,4.6e-4)	(9.7e-2,4.6e-4)

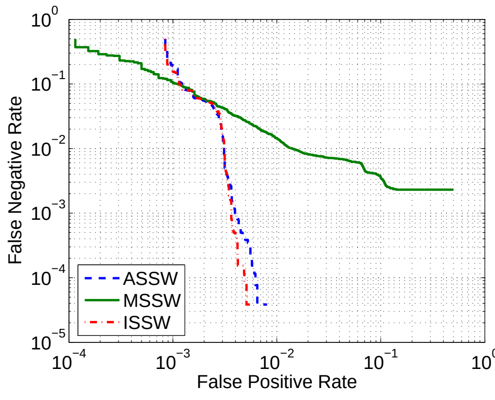
Table 3.7
The 1-AUC (area under the curve) results of three test algorithms (ASSW, MSSW, ISSW) for different attacks.

Attacks	Different Key	No Attack	Gaussian Filtering	Sharpening
ASSW	1.01e-3	1.09e-3	2.67e-2	1.33e-7
MSSW	3.19e-3	3.24e-3	6.24e-3	1.12e-3
ISSW	1.01e-3	1.11e-3	2.71e-2	4.28e-7

Attacks	JPEG	Amplitude Scaling	Histogram Equalization
ASSW	5.90e-2	3.64e-3	2.82e-5
MSSW	8.44e-3	3.70e-3	6.47e-3
ISSW	6.06e-2	3.68e-3	2.96e-5

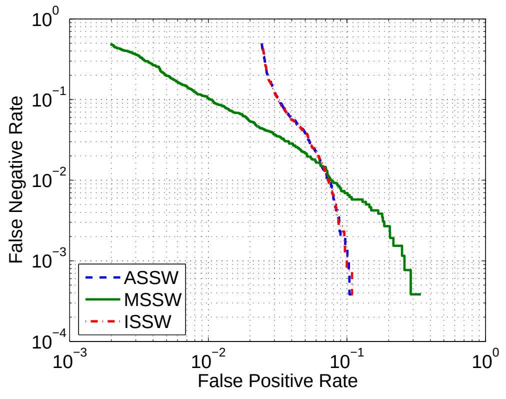


(a) Unwatermarked

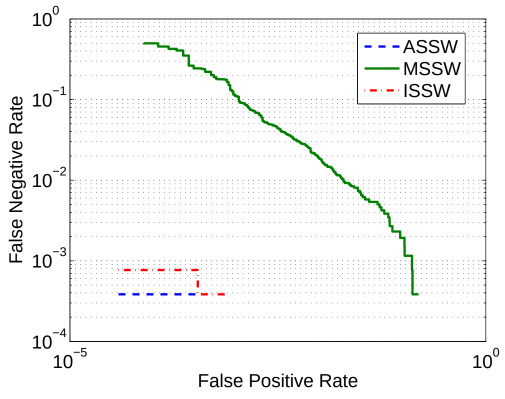


(b) Different key

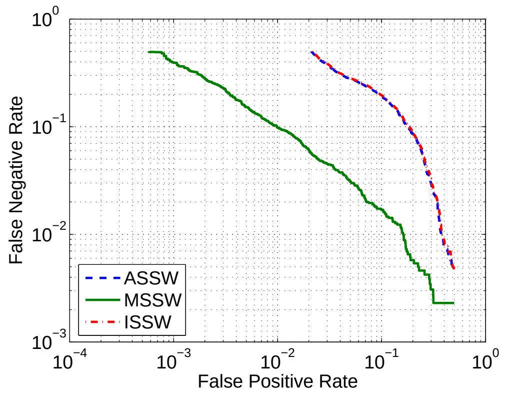
Fig. 3.6. The receiver operating characteristics (ROC) for three test algorithms (ASSW, MSSW, ISSW) without attacks, using false positive results from unwatermarked images (a) and different keys (b).



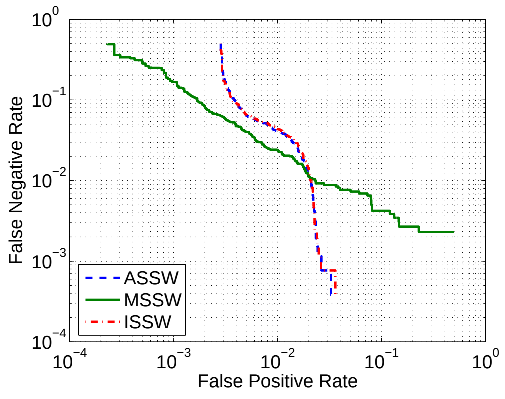
(a) Gaussian filtering



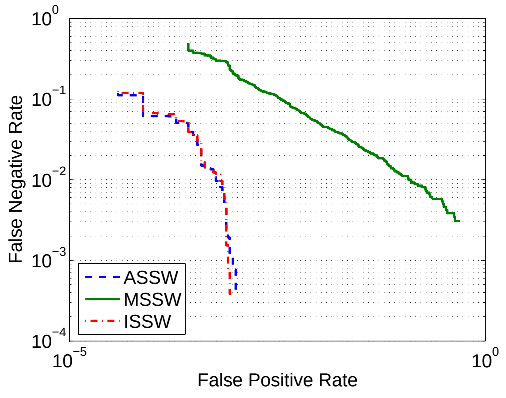
(b) Sharpening



(c) JPEG(Q=70)

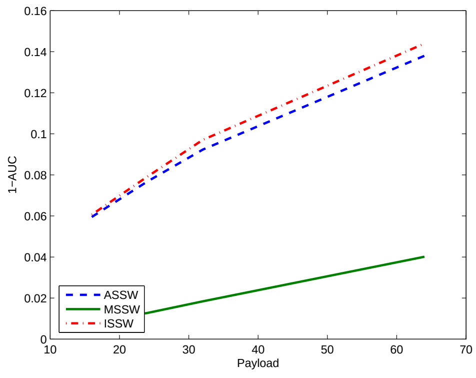


(d) Amplitude scaling

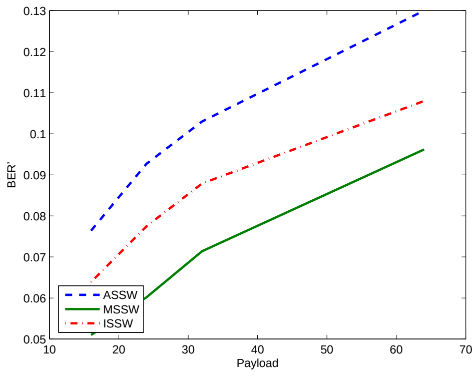


(e) Histogram equalization

Fig. 3.7. The receiver operating characteristics (ROC) for three test algorithms (ASSW, MSSW, ISSW) for different attacks.

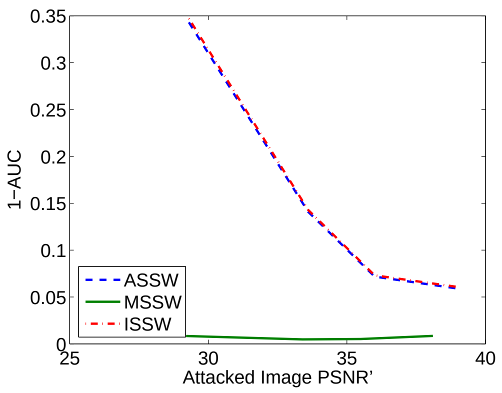


1-AUC

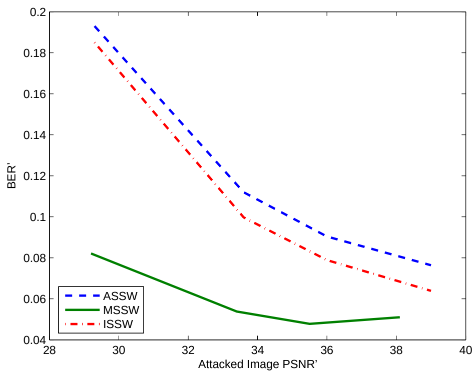


BER'

Fig. 3.8. The 1-AUC and BER' results of three test algorithms (ASSW, MSSW, ISSW) for different payloads for JPEG attack(Q=70).

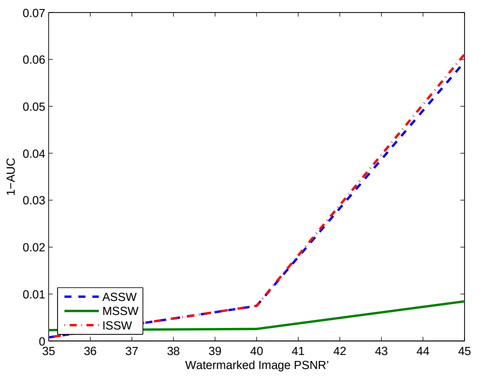


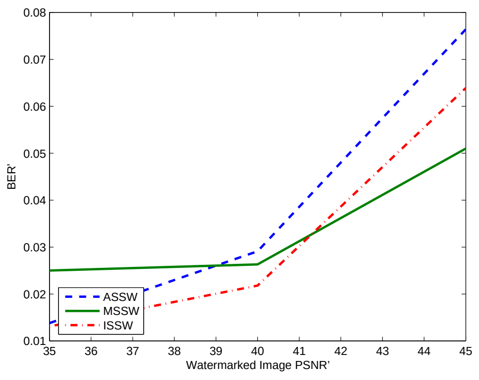
1-AUC



BER'

Fig. 3.9. The 1-AUC and BER' results of three test algorithms (ASSW, MSSW, ISSW) for different JPEG attacks (Q=10,30,50,70), indirectly represented as different attacked image PSNR' values.





1-AUC BER'

Fig. 3.10. The 1-AUC and BER' results of three test algorithms (ASSW, MSSW, ISSW) for JPEG attack(Q=70) with different embedding strengths, indirectly represented as different watermarked image PSNR'.

varying the embedding strength and payload and measuring the performance and fidelity for different attacks and images. We measured the performance using bit error rate (BER) and receiver operating characteristic (ROC) and fidelity using PSNR of the conditional mean image. One important property of evaluation is summarization and to summarize PSNR and BER results we used the Taguchi loss function which considers both the mean and variance. To summarize ROC, we used area under the curve (AUC). Even from the limited results we obtained, evaluation using actual images and attacks showed some characteristics of the implemented watermarking algorithms that would be hard to predict using theoretical models. For future work, we need a method to assign a robustness score to a watermarking technique based on the requirements of a particular application. One method is to assign different importance to the bit error rate, the false negative probability for the false positive probability required by the application and attacks and combine in them in a manner that is suitable to that application. To assign different importance, we would need a better understanding of the application requirements.

4. WATERMARK BENCHMARKING SYSTEM

4.1 Introduction

The definition from online Merriam-Webster Online Dictionary [96] states that a benchmark is "a standardized problem or test that serves as a basis for evaluation or comparison." There are widely used computer benchmarks used in the industry. A computer benchmark is typically a computer program that performs a strictly defined set of operations (a workload) and returns some form of result (a metric) describing how the tested computer performed. Computer benchmark metrics usually measure speed (how fast was the workload completed) or throughput (how many workloads per unit time were measured).

It is measured against a reference speed or throughput. Running the same computer benchmark on multiple computers allows a comparison to be made. Ideally, the best comparison test for systems would be your own application with your own workload. Unfortunately, it is often very difficult to get a wide base of reliable, repeatable and comparable measurements for comparisons of different systems on your own application with your own workload. This might be due to time, money, confidentiality, or other constraints.

There exists widely used benchmarks in the computer industry. SPEC [97] is an acronym for the Standard Performance Evaluation Corporation. SPEC designed SPEC CPU2000 to provide a comparative measure of compute intensive performance across the widest practical range of hardware. This resulted in source code benchmarks developed from real user applications. These benchmarks are dependent on the processor, memory and compiler on the tested system. It consists of 12 integer programs and 14 floating-point programs. The basic reason behind SPEC's success is that all parts of the market profit from having a technically well-respected and socially well-trusted “measuring stick” available that allows comparison of different systems for a certain purpose [61].

EEMBC [98], the Embedded Microprocessor Benchmark Consortium, was formed in 1997 to develop meaningful performance benchmarks for processors and compilers in embedded applications. EEMBC benchmarks have become an industry standard for evaluating the capabilities of embedded processors and compilers according to objective, clearly-defined, application-based criteria. It evaluates embedded processors in 6 areas: telecom, office automation, consumer, automotive/industrial, networking and microcontroller.

4.1.1 Current Still Image Watermark Benchmarks

As shown in chapter 1 watermark designers design watermark algorithms using various techniques. The role of benchmarking of watermarking is to provide a fair and automated evaluation of these techniques. While a significant portion of the community's effort has gone into proposing and in a few instances implementing attacks against different watermark algorithms, only StirMark [51, 52, 60] has been accepted by the community at large.

StirMark

StirMark [51, 52, 60] was originally written in 1997, and has been continually enhanced with new techniques. The first version of StirMark was published as a generic tool for simple robustness testing of image watermarking algorithms. It introduced random bilinear geometric distortions to de-synchronize watermarking algorithms. Then several versions followed improving the original attack but also introducing a longer lists of tests. One of the ambitions of StirMark is to provide a single tool that will be able to test different kinds of media such as images, sounds and videos. The project is being written using the C++ language to take full advantage of the inheritance and polymorphism features of an object-oriented language. For the Watermarking scheme functions, it uses a dynamic library provided by the user. You can write a evaluation profile in the form of Windows INI files.

Stirmark is currently developing a public automated web-based evaluation service for watermarking schemes [60, 99]. The features of the StirMark benchmark evaluation service are

- Simple interface with watermarking libraries.
- Takes into account the application of the watermarking scheme by proposing different evaluation profiles (tests and set of images) and strengths.
- Allows client to submit libraries for using Windows dll's.
- Ease of use: The client sends a library which follows a general interface to be evaluated and specifies the evaluation profile and level of assurance to be used. The StirMark Benchmark service automatically starts hundreds of tests on the library using its library of images. As soon as the test are finished the results are sent to the client and may later be published on the project website.
- All evaluation procedures, profiles and code are publicly available and available for download.

Certimark

The Certimark [61] Consortium consisted of 15 partners from European industry and academia. The consortium was developing a benchmark suite that will enable its users to evaluate digital watermarking technologies. The project started in May 2000 and ended in July 2002. Certimark benchmark uses XML schema as the format for all control and result information as they support the required flexibility and modularity with regard to the design of data structures. The Certimark benchmark supports still images and a limited set of professional-quality video clips. It runs on a web-based platform, with the server currently restricted to Windows NT/2000, while the client running the web browser can be any web browser. The objectives of Certimark [100] are

- To design, develop and publish a complete benchmark suite for watermarking technologies within promising application scenarios.
- To make this benchmark suite a reference tool both for technology suppliers and for technology customer within promising application scenarios.
- To set up a certification process for watermarking algorithms.
- To concentrate research on pending key issues in watermarking for protection of still pictures and low bit rate video over the Internet.

The benchmark can be presented as a suite of modules as shown in Figure 4.1, for which interfaces will be defined both towards the previous and the next module in the pipeline and to the operator of the benchmark. The modules in Certimark are defined as follows:

- Image source, delivering the content to be watermarked, according to categories of contents and to parameters defined for a particular benchmark session.
- SUT ("System Under Test") watermark embedder, provided by the tester, who can define some parameters that control SUT operation.
- Attack module, the first operative module in the benchmark, simulating all sorts of attacks on the watermark (intentional and non-intentional) resulting in possible loss of watermark readability. Can perform several sequential attacks.
- SUT watermark decoder, provided by the tester. Two main functions are achieved here, detection of the watermark and extraction of the payload for monitoring purposes.
- Comparator module, where the payload is compared to the original values inserted, according to parameters that are defined for the benchmarking purpose in relation with the monitoring tasks.

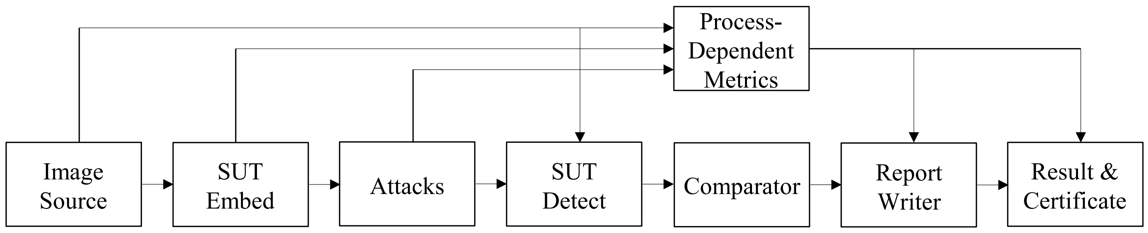


Fig. 4.1. The Certimark architecture for watermark evaluation.

- Process-dependent Metrics evaluation module, where metrics concerning visual quality, complexity, etc. are generated (raw data) and put into form (e.g., preliminary computations before plotting curves).
- Report writer is the module where all results are taken into account to write a benchmark report, with tables and graphics. This module is out of the run; it is called only once at the very end of a benchmarking session to compile the results.
- Result and Certificate is the module where Certimark knowledge is taken into account: results (curves. . .) are replaced between performance specifications of typical applications. This module is called after the report writer module.

Checkmark

Checkmark [44,54] is a benchmarking suite for digital watermarking technologies. Checkmark has many attacks that are not included in StirMark. The attacks include estimation based techniques to derive the optimal attacks for a given watermark distribution. The attacks in Checkmark not in StirMark include: denoising (ML and MAP), wavelet compression, watermark copy attack, active desynchronization, new geometrical attacks (projective transforms, collage attacks, random line removal) and copy attacks. Among 400 attacks in Checkmark, only 100 attacks are from

StirMark version 3.1. It also uses the Watson metric to measure the quality of images which is not included in StirMark. Running on MATLAB under UNIX and Windows, it provides efficient and effective tools to evaluate and rate watermarking technologies. Moreover, it takes the watermark application into account which means that the scores from individual attacks are weighted according to their importance for a given watermark usage. It uses Java for parsing XML files for the generation of applications and parsing of results. Currently, Checkmark only evaluates false negative probabilities for every attacks.

Optimark

Optimark [62] can be installed in any machine running Windows and it does not provide the source code. Main features of Optimark are

- Graphical user interface.
- Detection and decoding performance evaluation using multiple trials utilizing different watermarking keys and messages.
- Evaluation of the following detection performance metrics: For watermark detectors that provide a float output, i.e., the value of the test statistic used for detection:
  - Receiver Operating Characteristic curves (ROC), i.e. plots of the probability of false alarm versus the probability of false rejection.
  - Probability of false positive for a fixed, user defined, probability of false negative.
  - Probability of false negative for a fixed, user defined, probability of false positive.

- For watermark detectors that provide a binary output, i.e. a value that states whether the watermark has been detected or not: Probabilities of false positive and false negative.
- Evaluation of the following decoding performance metrics, for algorithms that allow for message encoding (multiple bit algorithms): Bit error rate, and percentage (probability) of perfectly decoded messages.
- Evaluation of the mean embedding and detection time.
- Evaluation of the data payload (for multiple bit algorithms).
- Evaluation of the algorithm breakdown limit for a certain attack and a certain performance criterion, i.e., evaluation of the attack severity where algorithm performance exceeds (or falls below) a certain limit.
- Result summarization in multiple levels using a set of user defined weights on the selected attacks and images.
- Option for both user defined and preset benchmarking sessions.

4.2 Watermark Evaluation Testbed

To facilitate the development of a universally adopted benchmark, we are developing at Purdue University a web-based system that will allow users to evaluate the performance of watermarking techniques. This system consists of reference software that includes both watermark embedders and watermark detectors, attack scenarios, evaluation modules and a large image database. The ultimate goal of the current work is to develop a platform that one can use to test the performance of watermarking methods and obtain fair, reproducible comparisons of the results. We feel that this work will greatly stimulate new research in watermarking and data hiding by allowing one to demonstrate how new techniques are moving forward the state of the art. We will refer to this system as the Watermark Evaluation Testbed or

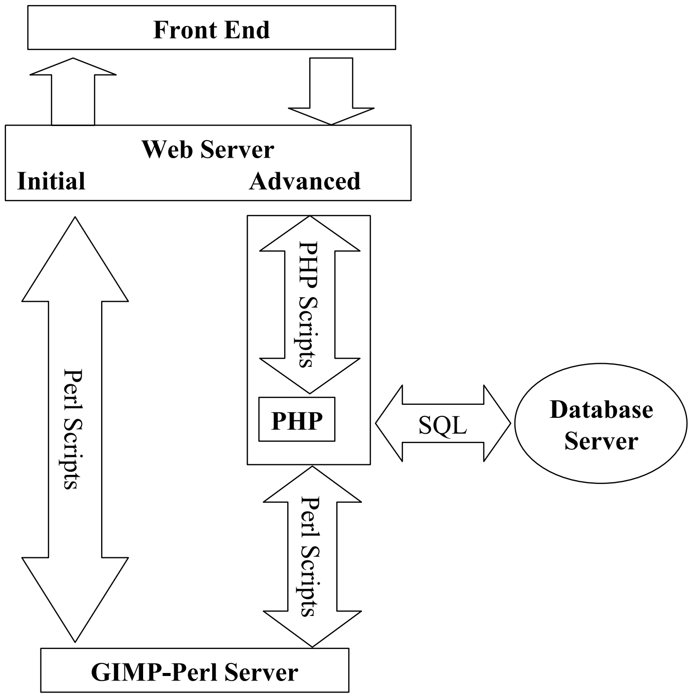


Fig. 4.2. The architecture of watermark evaluation testbed (*WET*).

WET [58, 69]. Architecture of the testbed is shown in Figure 4.2. *WET* consists of three major components: the front end, the algorithm modules, and the image database. Each component will be described below. Currently, *WET* runs on a 2.4 GHz Xeon dual processor computer using the Fedora 2.0 operating system[1].

4.2.1 Watermarking Benchmark Considerations

To design a watermarking evaluation system we have to consider the users, what we are going to evaluate, and how we are going to show the results.

[1]The system is located at http://www.datahiding.org. To obtain access to use *WET* contact wetbug@ecn.purdue.edu.

Benchmark Users

It is useful to make a distinction with regard to its users. They have somewhat different requirements as to what the benchmark should deliver, in particular in the analysis, reporting and certification of results.

Choice Makers

These users employ the benchmark to compare a set of watermarking algorithms for their appropriateness in a certain application scenario. Thus, the appropriate metric are derived from satisfying application requirements, and certified results may be required if a choice maker does not want to run the benchmark himself. From this point of view, the platform must provide a GUI that allows herself to specify the needs and requirements. To avoid user confusion, templates of typical applications should be provided, and the user must be also able to modify only those parameters he wants to study.

Developers

Developers use the benchmark to compare different versions of an algorithm, or different settings of its parameters, for improvements of its performance: robustness, error rates, imperceptibility, or execution speed. To this class of users, the internal interface of the benchmark is important, as they need to insert their watermarking algorithm into the platform. Thus, this interface should be as simple and as easy to use as possible. Once a watermarking algorithm has been successfully integrated into the platform, a developer might use it in a similar way to a choice maker, performing various runs with different image, different parameters and different versions of the watermarking algorithm, or he may use the raw results to perform tailored analysis supporting his development goals.

Students

Student is someone new to watermarking and wants to experiment with watermarking techniques. This user will access the basic reference software and evaluation tools to learn about watermarking.

Benchmark Design

The benchmark design has to manage the benchmark's complexity, and provide room for expansion, especially with a view to support watermarking algorithms for new types of multimedia data. Because a benchmark has to simulate a complete watermarking chain as shown in Figure 1.1, we can assume that data will pass through this chain more than once, with various parameters. The data flow within the benchmark can therefore considered to be a pipeline. Nevertheless, as the user needs to supply parts of this pipeline (i.e., the watermarking algorithms that are to be tested), we need to make the benchmark modular so that pieces can be easily exchanged. Every basic operation that is part of the classic watermarking chain needs to be separated from the others into one isolated module. These modules will obviously comply with a general detailed interface. Modularity provides several crucial advantages: modules can be exchanged easily; given well-defined interfaces, they can be developed separately, and they can be upgraded when needed. Although this flexibility allows a good understanding of every separate part of the benchmark, one has to make them interface correctly together with some sort consistency all along the benchmarking process. This implies that every module has to conform to certain rules for interoperability, and that the benchmark platform needs to guarantee control and integrity of the benchmark as a whole.

Evaluation Profile

The input of a benchmarking system is the watermark embedding and the detection-decoding software and its outputs are performance values and plots that illustrate the performance of the watermarking system under test against various attacks. The benchmarking system's parameters are

- Image data set: The images that should be used in the benchmarking system should vary in size and frequency content, since these two factors affect the watermarking system performance. Moreover, the types of images in the image data set (indoor/outdoor scenes, graphic images etc.) should be consistent with those that can be met in a real world application scenario. A fair benchmark should use a wide range of picture sizes and different types of images. For fair comparison, the same set of images should be used for different watermarking algorithms.

- Keys and messages data set: The number of keys used in the benchmark is a very important issue. The reason is that in several watermarking methods the algorithm's performance depends on the watermark key and the message. As we use more keys and messages to evaluate an algorithm, we can reduce the variability in our performance measurements.

- Fidelity: In order to rate the watermark visibility and the perceptual quality of the watermarked image an objective measure should be used.

- Attacks: The attacks should include all attacks that are required by many applications. It should include all distortions caused by normal image usage, transmission, storage, etc.

Benchmarking Protocol Modules

The benchmarking system comprises the watermark embedding module, the attack module, the watermark detection-message decoding module and the performance evaluation module.

- Image database: A requirement for a watermark benchmark is to have a image database. It is important to test an image watermarking software on many different images and for fair comparison the same set of sample images should always be used. The images should also cover a broad range of contents, types and sizes.
- User interface: this will be the "window" into the system for users. Users will be required to identify themselves (no anonymous use will be permitted) but very little information about a user will be stored in the system. For example, which components a given user has exercised will not be maintained. The user interface will allow a researcher to chose several standard evaluation profiles or develop their own profile. The user interface will also supply to the user the results of the tests.
- Submission Interface: This will allow users to submit components that they have designed and wish to have evaluated.
- Embedding module: The watermark embedding module utilizes the embedding software and takes as inputs the image, quality, key and message sets.
- Attack module: The attack module is used to distort the watermarked images that have been generated in the watermark embedding stage. It can perform several sequential attacks.
- Performance evaluation module: The performance evaluation module is used in order to extract the performance scores/plots of the watermarking algorithm under test. It measures fidelity, costs, false negative and false positive rates.

It could produce Receiver Operating Characteristic (ROC) curve based on the false negative and positive rates.

4.2.2 Front End

Overview

The Front End is the end users' main interface into *WET*. The user interface abstracts much of the underlying architecture and allows the user to focus on the tasks to be performed. The Front End consists of three major components: the web server, the database server, and the GIMP-Perl server. It provides a web interface whereby a user can select various tasks to be performed. These tasks include selecting images to be watermarked, attacking a watermarked image, or detecting the presence of a watermark in a particular image. It also has an administrative interface. The administrative interface is provided to allow and administrator to perform various tasks as image database management and security functions. Two versions of *WET*, the *initial version* and *advanced version* are available.

The *initial version* provides a static image database of about fifty images and a intuitive user interface. It is intended for either first time users to familiarize themselves with the system or for users who want to get a feel for how watermarking works. Several watermarking algorithms and attack algorithms are available to the user. A basic operations in the *initial version* are illustrated in Figure 4.3.The user selects a single image from the available images, watermarks the image, optionally attacks the watermarked image, and can detect the presence of a watermark in the watermarked (and possibly attacked) image. Several statistics including the CPU time used for embedding/detecting the watermark, the mean square error (MSE) between the original and watermarked image, the difference image between the watermarked and original image, and some algorithm specific statistics such as the correlation value are provided as part of the result of these steps.

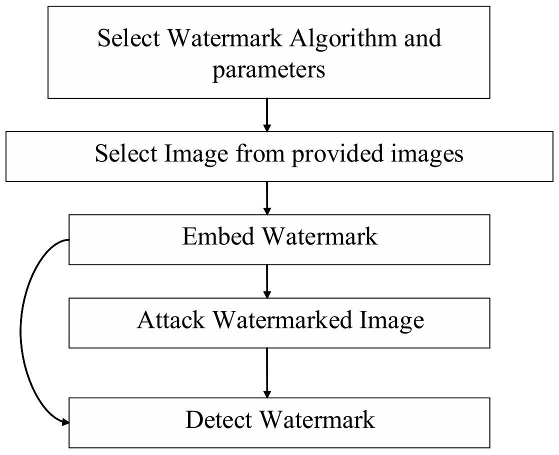

Select Watermark Algorithm and parameters
Select Image from provided images
Embed Watermark
Attack Watermarked Image
Detect Watermark

Fig. 4.3. Watermarking procedure for the initial version in *WET*.

The *advanced version* provides an extensive image database. The user can select images with particular attributes, i.e. chrominance, resolution, height, width, category, etc., to work with. All metrics reported in the *initial version* are also reported in the *advanced version.* The *advanced version* is available in two modes: interactive and batch modes. The interactive mode allows the user to manually step through the process illustrated in Figure 4.4. The process of the interactive mode is similar to the *initial version* except that it can select up to 25 images, and multiple attacks can be performed on the same watermarked image. The interaction between all components that implements the interactive mode is illustrated in Figure 4.5. The batch mode allows for user submitted jobs. The user selects the images, the watermarking algorithm and corresponding parameters, the attack algorithms (in order of which they should be performed), and the detection(s) to be performed. The user created job is executed and the results are sent back to the user upon completion of the tasks in the for of email.

Software Implementation

The programming languages used to implement the Front End are:

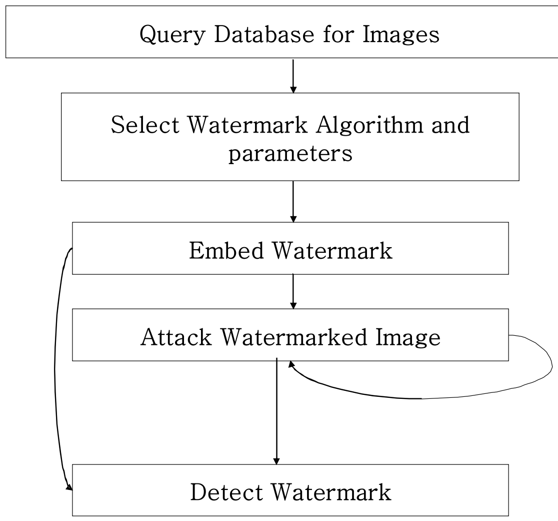

Query Database for Images
Select Watermark Algorithm and parameters
Embed Watermark
Attack Watermarked Image
Detect Watermark

Fig. 4.4. Watermarking procedure for the advanced version in *WET*.

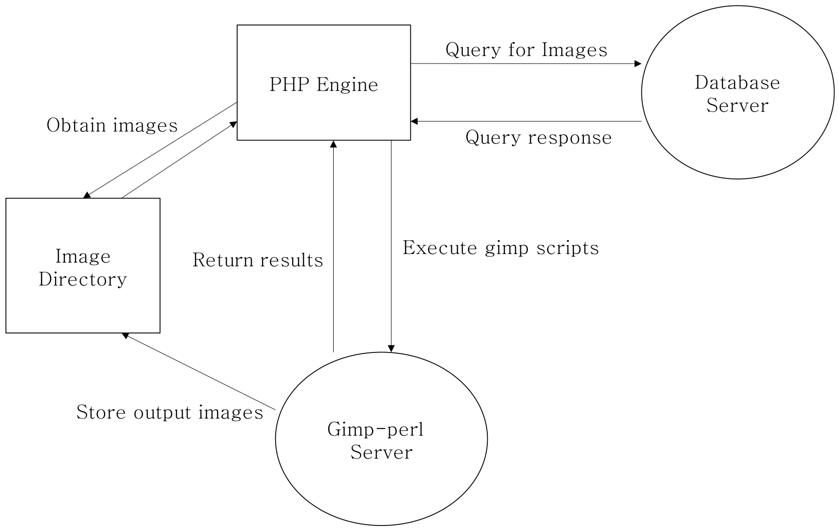


Fig. 4.5. Interactions of major components in the advanced version of *WET*.

JavaScript Used for validating user input before sending the request from the web browser. It is also used to improve the user interface to the available options.

Perl Used for scripts that communicate with GIMP to execute the available algorithms available in *WET* and report the result to the user. It is also used for scripts that perform administrative tasks such as restarting *WET* and generating image thumbnails.

HTML Used for providing the static components of the user interface.

SQL Used for interacting with the database server to select images matching user specified criteria.

PHP Used extensively in the *advanced version.* It communicates with the Database Server, the Perl Scripts, and also generates pages to present results from other components to the user. It is used to dynamically generate Perl scripts to perform the tasks selected in the batch mode.

The MySQL database engine is used as the database server [101]. It maintains the attributes of all images available in *WET*. We use PHP to interface with the database server. Using PHP, we send queries and receive responses from the MySQL server.

The GIMP-Perl server is an add-on to the GNU Image Manipulation Program (GIMP) [102] that performs batch mode image manipulations. It takes programs written in Perl and executes them using GIMP and returns the result to the Perl interpreter. We use the GIMP-Perl server to run Perl scripts that execute GIMP plug-ins.

All the scripts used for the *initial version* are implemented in Perl. Each algorithm requires an image and specific parameters from the user. The images and parameters are selected on the user interface and passed to web server. The web server in turn calls the appropriate Perl script with the parameters. All Perl scripts used for embedding a watermark, detecting a watermark, or attacking a watermarked image have similar structures because algorithms are implemented as GIMP plug-ins [102].

A similar approach is used for the interactive mode of the *advanced version* except that the PHP engine acts as an intermediary between the web server and the Perl scripts. Once the user selects the images to watermark along with the corresponding parameters, the request is sent to the PHP engine via the web server and the PHP engine executes the appropriate Perl scripts. The results from the Perl scripts are passed back to the PHP engine. Upon completion of the Perl script, the PHP engine generates a page with the results and sends it to the web server for onward display to the user.

In the batch mode, the user selects the algorithms, images, other tasks to be performed, and the correct order of execution and then submits the request to the web server. The web server in turn forwards the request to the PHP Engine. The PHP Engine creates a Perl script by using several functions that generate the required code to perform the specific algorithms. Upon completion of the script generation, the PHP Engine executes the scripts. Currently, the results obtained upon job completion are sent to the user in the form of an electronic mail.

4.2.3 Algorithm Modules

It is desirable to develop tools that users can use standalone in their own test environments, allowing them to validate their tests locally before submitting them to a watermark benchmark site. To achieve this, the GNU Image Processing Program (GIMP) [102] was selected for use in *WET*. GIMP is designed to be augmented with plug-ins and extensions. Additionally, it provides full scripting support in various languages (Scheme, Perl, Java, Python, etc.). The advanced scripting interface of GIMP allows everything from the simplest task to the most complex image manipulation procedures to be easily scripted. The combination of extensibility and scripting support make GIMP a powerful environment for *WET*. The GIMP community also provides a large selection of plug-ins, which by their nature (geometric distortion, difference image) could be used for attack or measurement components of *WET*.

As previously mentioned, by allowing users to duplicate the test environment locally in a form they are familiar with, there is no distinction between their local development components and the ones needed for *WET*. Stated another way, an Experimenter can natively develop in GIMP and after they are ready for testing, they can submit the same binary to a watermark benchmark site they used in their local development. This eliminates the step of packaging the component for an external benchmark and never knowing where a bug was introduced.

We have been implementing several plug-ins for GIMP. Our plug-ins can be used in two different modes: Interactive Mode and Non-Interactive Mode. The former has a user interface where the user can choose the input parameters and the output parameters are displayed after using the plug-in. The Non-Interactive Mode performs the same function as the Interactive Mode but allows the plug-in to be called from another GIMP plug-in or scripts. The Non-Interactive Mode is used in *WET*.

Watermarking Algorithms

To extend watermarking algorithms to color images, we used the reversible color transform (RCT) [103] developed for JPEG2000. The Red, Green and Blue components of an image are transformed by RCT and we embed the watermark into the luminance component. The RCT is shown in Figure 4.6. We chose RCT among different color transforms because it preserves the image when a watermark is not embedded. We watermarked the luminance component because a watermark should be placed in the perceptually most significant components of an image [1].

One of the performance measures of a watermarking algorithm is computational complexity. We currently measure the computational complexity in terms of CPU execution time. Our GIMP plug-ins are written such that they return the CPU execution time as an output parameter. The watermarking algorithms we implemented as a GIMP plug-in include:

Secure spread spectrum watermarking [1]: This is a multiplicative spread spectrum watermarking algorithm and is based on the assumption that the watermark should be placed in the perceptually most significant components of an image.

Blind wavelet watermarking [104]: This is a blind wavelet watermarking algorithm. A typical problem with spread spectrum watermarking is that the order and the number of significant coefficients can change due to various image manipulations. This technique addresses this problem by using two different thresholds to embed and detect the watermark.

Semi-Fragile watermarking [19]: This is a semi-fragile watermarking method that can detect information altering transformations even after the watermarked content is subjected to information preserving alterations. It is also capable of tolerating some degree of change to the watermarked image.

Quantization Index Modulation (QIM) watermarking [105, 106]: QIM embeds a message by using an ensemble of quantizers. Our algorithm is based on low complexity method that is known as dither modulation.

LOT Based Adaptive Watermark [107] The LOT watermark algorithm embeds an invisible, robust watermark in an image using the spread spectrum scheme. The Lapped Orthogonal Transform (LOT) is chosen for the block-based orthogonal transform, instead of DCT, to avoid blocking effects. Moreover, the Human Visual System (HVS) properties are exploited to adjust the intensity of our embedded watermark according to the local features of the image.

Locktography [108] Locktography is a steganography technique which embeds data into a color still image by modifying the least-significant bits (LSB) of the image.

$$\begin{aligned} Y &= \lfloor \frac{R+2G+B}{4} \rfloor \\ Dr &= R-G \\ Db &= B-G \end{aligned}$$

(a) Forward RCT

$$\begin{aligned} G &= Y - \lfloor \frac{Dr+Db}{4} \rfloor \\ R &= Dr+G \\ B &= Db+G \end{aligned}$$

(b) Inverse RCT

Fig. 4.6. Reversible Color Transform (RCT).

StirMark 3.1

StirMark 3.1 is from [51, 52, 109]. We implemented the default test mode in StirMark 3.1 as a GIMP plug-in. StirMark 3.1 implements the following possible attacks on a watermarked image: linear filtering, median filtering, Frequency Mode Laplacian Removal (FMLR), JPEG compression, color quantization, scaling, shear, aspect ratio, general linear transform, rotate and crop, rotate and scale, cropping, flip, remove row and column and original StirMark attack (random bilinear geometrical distortion).

StirMark 4.0

In addition to StirMark [51, 52, 89] 3.1 attacks we implemented the StirMark 4.0 attacks as separate GIMP plug-ins. The attacks are classified into geometric transforms, signal processing operations and special transforms. Geometric transforms include affine transform, rescale, rotation and small random distortions. Signal processing attacks operations include adding dither noise, adding uniformly distributed noise, filtering by convolution, median cut and self-similarities attack. Special Transforms include flipping(horizontal, diagonal, and vertical), cropping, remove lines, and special rotations (90, 180, and 270 degrees).

Mean Square Error Module

One important performance measure is fidelity. Fidelity is the perceptual similarity between the original image and the watermarked image [5]. We currently measure the fidelity in terms of Mean Square Error (MSE). Mean Square Error is obtained as follows

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(p(i,j) - \hat{p}(i,j)\right)^2$$

where M is the number of color components, N is the number of pixels, p is the original image and $\hat{p}$ is the modified image. We implemented a plug-in that takes two same size images as input, and outputs the mean square error and the difference image.

4.2.4 Image Database

WET use MySQL [101] as the database engine. The image database maintains the attributes of all images available in the test-bed. We currently have 1301 images which are copyright free. The images are from variety of cameras and sensors including scanned photographs, x-ray images, ultrasound images, astrometrical images, line drawings, digital cameras, maps, and computer generated images. Each image in the database is stored with its attributes including chrominance, resolution, height, width, and category. A sample of the images are shown in Figure 4.7. The image size range from 120x120 thumbnail size images to 3848x5768 mammogram images.

4.3 Conclusion

While digital watermarking has received much attention within the academic community and private sector in recent years, it is still a relatively young technology. As such there are few widely accepted benchmarks that can be used to validate the performance claims asserted by members of the research community. This lack of a universally adopted benchmark has hindered research and created confusion within the general public. To facilitate the development of a universally adopted benchmark, we are developing at Purdue University a web-based system that will allow users to evaluate the performance of watermarking techniques. We refer to this system as the Watermark Evaluation Testbed or *WET* [58, 69]. This system consists of reference software that includes both watermark embedders and watermark detectors, attack scenarios, evaluation modules and a large image database. The ultimate goal of the current work is to develop a platform that one can use to test the performance of watermarking methods and obtain fair, reproducible comparisons of the results. We

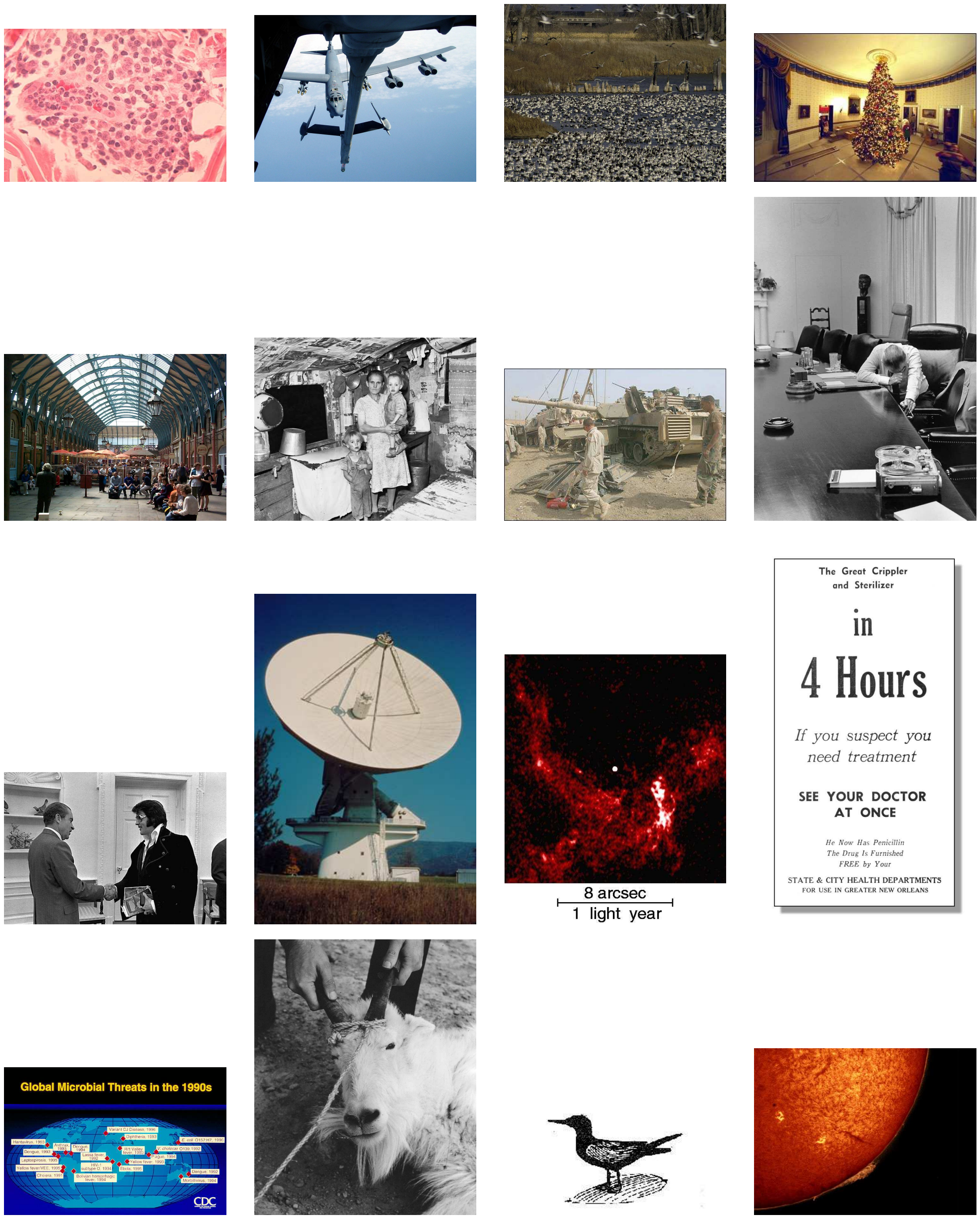


Fig. 4.7. Sample images from the *WET* image database.

feel that this work will greatly stimulate new research in watermarking and data hiding by allowing one to demonstrate how new techniques are moving forward the state of the art.

5. CONCLUSIONS

Watermarks has many potential applications. One important part in facilitating the wide adoption of watermarking is performance evaluation or benchmarking. The role of benchmarking of watermarking is to provide a fair and automated evaluation of important properties in watermarks such as robustness and fidelity. In this dissertation, we investigated still image watermark evaluation in terms of fidelity and robustness evaluation and design of a web based watermark benchmark system.

5.1 Contributions of this Dissertation

- **Mean square error fidelity metric for presentation attacked images**

  Evaluating fidelity of the attacked images is important for attack development and consequently watermark development. Presentation attacks are attacks that does not remove the watermark but makes the watermark undetectable by changing how it is presented to detector. We addressed the fidelity evaluation of special cases of presentation attacks. We described a technique to measure fidelity for presentation attacked images in terms of mean square error by compensating these attacks. We compensate these attacks by using the properties of the human visual system (HVS). Valumetric attacks are monotonic increasing or decreasing function of pixel values and include contrast enhancement and gamma correction. Since the nonlinearity does not change the content of the image, we measure fidelity by compensating the attack using conditional mean. We proved that conditional mean becomes a monotonically increasing or decreasing function when used to compensate valumetric attacks. For pixel loss attacks such as cropping or jitter attacks, the human visual system (HVS) can interpolate and extrapolate the lost pixels. We implemented this property using error concealment developed in the video transmission literature [81]. For geometrical attacks, the human visual system (HVS) is not affected much for minor geometrical distortions. To compensate geometrical attacks, instead of using image registration techniques [57], we approximated the mapping from the original image to the attacked image using the information available for better image registration. This is true in a watermarking evaluation framework.

- **New summary statistics for still image watermark evaluation**

  Current watermarking benchmarks use summary statistics to summarize the results. One example is the average. For example, average bit error rate is used to summarize the bit error rates and average PSNR is used to summarize the PSNR values. This does not reflect still watermarking scenarios where each watermarked image is distributed to different people where the distribution of the results become important.

  Another approach to summarization is to use a threshold. We could measure the maximum bit error rate and compare it against a threshold. For fidelity, we could compare the fidelity results against the just noticeable difference (JND) [5, 6] value. But requiring the watermarking system to meet a hard threshold requirement puts too much restriction on the watermarking system.

  In quality engineering [66, 67], it is recommended that we should also consider the variance of the distribution as well as the mean. We propose a summary statistic for PSNR and BER using the Taguchi loss function [66, 67] which considers both the mean and variance. We compare 3 watermarking algorithms using the new summary statistics.

- **Watermark Evaluation Testbed**

  While digital watermarking has received much attention within the academic community and private sector in recent years, it is still a relatively young technology. As such there are few widely accepted benchmarks that can be used to validate the performance claims asserted by members of the research community. This lack of a universally adopted benchmark has hindered research and created confusion within the general public. To facilitate the development of a universally adopted benchmark, we are developing at Purdue University a web-based system that will allow users to evaluate the performance of watermarking techniques. We refer to this system as the Watermark Evaluation Testbed or *WET* [58, 69]. This system consists of reference software that includes both watermark embedders and watermark detectors, attack scenarios, evaluation modules and a large image database. The ultimate goal of the current work is to develop a platform that one can use to test the performance of watermarking methods and obtain fair, reproducible comparisons of the results. We feel that this work will greatly stimulate new research in watermarking and data hiding by allowing one to demonstrate how new techniques are moving forward the state of the art.

5.2 Future Work

The future work can be explored from the following three perspectives:

- For evaluation of valumetric attacks, we used conditional mean to compensate the attack. We need to extend conditional mean to consider the frequency component of the image or correlation between adjacent pixels. For error concealment, we need to employ a better technique than the neighborhood mean. We could use multiresolution error concealment techniques to improve fidelity. For forward mapping of the bilinear transform, we need to improve speed of the mapping. We could use information from previous pixels to improve speed.
- We used bit error rate and receiver operating characteristic (ROC) for the evaluation of a watermarking system. When the probability of error of a watermarking system is claimed to be 10^{-12}, it is almost impossible to check

this claim because we would have to have more than 10^{12} images or try more than 10^{12} different keys. We need a better way to extrapolate the probabilities of error. Bit errors can be modeled by characterizing the transmission channel [41]. There has been work in probability estimation in digital watermarking by modeling the watermarking process. Hernandez et al. [110, 111] gives a tighter bound than the Chernoff bound using the central limit theorem arguments using Gaussian and generalized Gaussian models. Miller et al. [112] predicts the false positive probability using the spherical method.

- In this dissertation, we only considered the evaluation of robust watermarks. Authentication watermarks have different properties than robust watermarks. Important properties of authentication watermarks is localization.

LIST OF REFERENCES

LIST OF REFERENCES

[1] I. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673–1687, 1997.

[2] J. Fridrich and M. Goljan, "Comparing robustness of watermarking techniques," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, vol. 3657, (San Jose, CA), pp. 214–225, Jan. 1999.

[3] R. B. Wolfgang, C. I. Podilchuk, and E. J. Delp, "Perceptual watermarks for digital images and video," *Proceedings of the IEEE*, vol. 87, pp. 1108–1126, Jul. 1999.

[4] C. I. Podilchuk and E. J. Delp, "Digital watermarking: Algorithms and applications," *IEEE Signal Processing Magazine*, vol. 18, pp. 33–46, Jul. 2001.

[5] I. Cox, M. Miller, and J. Bloom, *Digital Watermarking.* Morgan Kaufmann, 2001.

[6] M. Barni and F. Bartolini, *Watermarking Systems Engineering.* Marcel Dekker, Inc., 2004.

[7] E. T. Lin, A. M. Eskicioglu, R. L. Lagendijk, and E. J. Delp, "Advances in digital video content protection," *Proceedings of the IEEE*, vol. 93, pp. 171–183, January 2005.

[8] A. M. Eskicioglu and E. J. Delp, "An overview of multimedia content protection in consumer electronics devices," *Signal Processing: Image Communication*, vol. 16, pp. 681–699, 2001.

[9] A. M. Eskicioglu, J. Town, and E. J. Delp, "Security of digital entertainment content from creation to consumption," *Signal Processing: Image Communication*, vol. 18, pp. 237–262, 2003.

[10] B. Schneier, *Applied Cryptography.* New York: John Wiley and Sons, second ed., 1996.

[11] E. T. Lin, C. I. Podilchuk, T. Kalker, and E. J. Delp, "Streaming video and rate scalable compression: What are the challenges for watermarking?," *Journal of Electronic Imaging*, vol. 13, pp. 198–205, Jan. 2004.

[12] I. J. Cox and M. L. Miller, "The first 50 years of electronic watermarking," *EURASIP Journal of Applied Signal Processing*, vol. 2002, no. 2, pp. 126–132, 2002.

[13] I. J. Cox, M. L. Miller, and J. A. Bloom, "Watermarking applications and their properties," *International Conference on Information Technology: Coding and Computing*, pp. 6–10, 2000.

[14] F. Mintzer, G. Braudaway, and M. Yeung, "Effective and ineffective digital watermarks," *Proceedings of the IEEE International Conference on Image Processing*, (Santa Barbara, CA), pp. 9–12, Oct. 1997.

[15] E. J. Delp, "Is your document safe: An overview of document and print security," *NIP18: International Conference on Digital Printing Technologies*, (San Diego, CA), Sep. 29–Oct. 4 2002. presented at.

[16] S. Craver, N. Memon, B. Yeo, and M. M. Yeung, "Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 573–586, May 1998.

[17] E. T. Lin and E. J. Delp, "A review of fragile image watermarks," *Proceedings of the Multimedia and Security Workshop (ACM Multimedia '99)*, (Orlando, Florida), pp. 25–29, Oct. 1999.

[18] J. Fridrich, "Methods for tamper detection in digital images," *Proceedings of the Multimedia and Security Workshop (ACM Multimedia '99)*, (Orlando, Florida), pp. 19–23, Oct. 1999.

[19] E. T. Lin, C. I. Podilchuk, and E. J. Delp, "Detection of image alterations using semi-fragile watermarks," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, vol. 3971, (San Jose, CA), pp. 152–163, Jan. 2000.

[20] G. Friedman, "The trustworthy digital camera: Restoring credibility to the photographic image," *IEEE Transactions on Consumer Electronics*, vol. 39, pp. 905–910, Nov 1993.

[21] F. A. Petitcolas, R. Anderson, and M. Kuhn, "Information hiding-a survey," *Proceedings of the IEEE*, vol. 87, pp. 1062–1077, Jul 1999.

[22] A. K. Jain, *Fundamentals of digital image processing.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1989.

[23] J. J. K. O. Ruanaidh and T. Pun, "Rotation, scale and translation-invariant spread spectrum digital image watermarking," *Signal Processing*, vol. 66, no. 3, pp. 303–317, 1998.

[24] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 40–44, 1991.

[25] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard.* New York: Van Nostrand Reinhold Company, 1992.

[26] A. B. Watson, "DCT quantization matrices visually optimized for individual images," *SPIE Proceedings, Human Vision, Visual Processing and Digital Display IV*, vol. 1913, (San Jose, CA), pp. 202–216, Feb. 1993.

[27] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, “Image coding using wavelet transform,” *IEEE Transactions on Image Processing*, vol. 1, pp. 205–220, Feb. 1992.

[28] D. Knuth, *Seminumerical Algorithms*, vol. 1 of *The Art of Computer Programming*. Reading, Massachusetts: Addison-Wesley, third ed., 1997.

[29] *Information hiding techniques for steganography and digital watermarking*. Artech House computer security series, Norwood, MA: Artech House Inc., 2000.

[30] J. Su and B. Girod, “Power-spectrum condition for energy-efficient watermarking,” *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, (Kobe, Japan), pp. 301–305, Oct. 1999.

[31] C. Podilchuk and W. Zeng, “Image-adaptive watermarking using visual models,” *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 525–539, May 1998.

[32] M. Swanson, M. Kobayashi, and A. Tewfik, “Multimedia data-embedding and watermarking technologies,” *Proceedings of the IEEE*, vol. 86, pp. 1064–1087, Jun 1998.

[33] N. Jayant, J. Johnston, and R. Safranek, “Signal compression based on models of human perception,” *Proceedings of the IEEE*, vol. 81, pp. 1385–1420, Oct. 1993.

[34] B. Zhu, A. Tewfik, and O. Gerek, “Low bit rate near-transparent image coding,” *Proceedings of the SPIE International Conference on Wavelet Applications for Dual Use*, vol. 2491, (Orlando, Florida), pp. 173–184, 1995.

[35] B. Hannigan, A. Reed, and B. Bradley, “Digital watermarking using improved human visual system model,” *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, vol. 4314, (San Jose, CA), pp. 468–474, Jan. 2001.

[36] M. Costa, “Writing on dirty paper,” *IEEE Transactions on Information Theory*, vol. 29, pp. 439–441, May 1983.

[37] I. J. Cox, M. L. Miller, and A. McKellips, “Watermarking as communications with side information,” *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1127–1141, 1999.

[38] B. Chen and G. W. Wornell, “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.

[39] B. Chen and G. W. Wornell, “An information-theoretic approach to the design of robust digital watermarking systems,” *IEEE Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2061–2064, 1999.

[40] S. Voloshynovskiy, A. Herrigel, N. Baumgärtner, and T. Pun, “A stochastic approach to content adaptive digital image watermarking,” *International Workshop on Information Hiding*, vol. 1768 of *Lecture Notes in Computer Science*, (Dresden, Germany), pp. 212–236, Springer Verlag, 29 Sep. –1 Oct. 1999.

[41] J. G. Proakis, *Digital Communications.* McGraw-Hill, third ed., 1995.

[42] M. Kutter and F. A. P. Petitcolas, “A fair benchmark for image watermarking systems,” *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, vol. 3657, (San Jose, CA), pp. 226–239, Jan. 1999.

[43] P. Meerwald, “Digital image watermarking in the wavelet transform domain,” Master’s thesis, Department of Scientific Computing, University of Salzburg, Austria, Jan. 2001.

[44] S. Voloshynovskiy, S. Pereira, T. Pun, J. Eggers, and J. Su, “Attacks on digital watermarks: Classification, estimation-based attacks and benchmarks,” *IEEE Communications Magazine*, vol. 39, no. 8, pp. 118–127, 2001.

[45] S. Voloshynovskiy, S. Pereira, V. Iquise, and T. Pun, “Attack modelling: Towards a second generation watermarking benchmark,” *Signal Processing*, vol. 81, pp. 1177–1214, June 2001.

[46] A. K. Jain, *Fundamentals of Digital Image Processing.* New Jersey: Prentice-Hall, first ed., 1989.

[47] S. Kalluri, *Nonlinear Adaptive Algorithms for Robust Signal Processing and Communications in Impulsive Environments.* PhD thesis, Department of Electrical and Computer Engineering, University of Delaware, Newark, DE, Dec. 1998.

[48] D. L. Donoho, “De-noising by soft-thresholding,” *IEEE Transactions on Information Theory*, vol. 41, pp. 613–627, May 1995.

[49] S. Craver, B.-L. Yeo, and M. Yeung, “Technical trials and legal tribulations,” *Commun. ACM*, vol. 41, no. 7, pp. 45–54, 1998.

[50] E. T. Lin and E. J. Delp, “Temporal synchronization in video watermarking,” *IEEE Transactions on Signal Processing*, vol. 52, pp. 3007–3022, Oct. 2004.

[51] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, “Attacks on copyright marking systems,” *Information Hiding, Second International Workshop*, (Portland, OR), pp. 219–239, Springer-Verlag, Apr. 1998.

[52] F. A. P. Petitcolas, “Watermarking schemes evaluation,” *IEEE Signal Processing Magazine*, vol. 17, pp. 58–64, Sep. 2000.

[53] F. Hartung and M. Kutter, “Multimedia watermarking techniques,” *Proceedings of the IEEE*, vol. 87, pp. 1079–1107, Jul 1999.

[54] S. Pereira, S. Voloshynovskiy, M. Madueño, S. Marchand-Maillet, and T. Pun, “Second generation benchmarking and application oriented evaluation,” *Information Hiding Workshop*, (Pittsburgh, PA), Apr. 2001.

[55] P. Moulin and R. Koetter, “Data-hiding codes,” *to appear in Proceedings IEEE*, December 2005.

[56] M. Kutter and F. A. P. Petitcolas, “Fair evaluation methods for image watermarking systems,” *Journal of Electronic Imaging*, vol. 9, pp. 445–455, Oct. 2000.

[57] B. Macq, J. Dittmann, and E. J. Delp, "Benchmarking of image watermarking algorithms for digital rights management," *Proceedings of the IEEE*, vol. 92, pp. 971–984, June 2004.

[58] H. C. Kim, E. T. Lin, O. Guitart, and E. J. Delp, "Further progress in Watermark Evaluation Testbed (WET)," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, Proceedings of SPIE Electronic Imaging, (San Jose, CA), pp. 241–251, Jan. 2005.

[59] S. Decker, "Engineering considerations in commercial watermarking," *IEEE Communications Magazine*, vol. 39, Aug. 2001.

[60] F. A. P. Petitcolas, M. Steinebach, F. Raynal, J. Dittman, C. Fontaine, and N. Fates, "A public automated web-based evaluation service for watermarking schemes: StirMark benchmark," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, vol. 4314, (San Jose, CA), Jan. 2001.

[61] J. C. Vorbruggen and F. Cayre, "The Certimark benchmark: architecture and future perspectives," *IEEE International Conference on Multimedia and Expo*, vol. 2, (Lausanne, Switzerland), pp. 485–488, Aug. 2002.

[62] V. Solachidis, A. Tefas, N. Nikolaidis, S. Tsekeridou, A. Nikolaidis, and P. Pitas, "A benchmarking protocol for watermarking methods," *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, (Thessaloniki, Greece), pp. 1023–1026, Oct. 2001.

[63] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, April 2004.

[64] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, (Montreal, Canada), pp. 709–712, May 2004.

[65] Z. Wang and E. P. Simoncelli, "Translation insensitive image similarity in complex wavelet domain," *Proceedings of the IEEE International Conference on Acoustics, Speech, & Signal Processing*, vol. 2, (Philadelphia, PA), pp. 573–576, March 2005.

[66] T. F. Rodriguez and D. A. Cushman, "Optimized selection of benchmark test parameters for image watermark algorithms based on Taguchi methods and corresponding influence on design decisions for real-world applications," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, vol. 5020, (San Jose, CA), Jan. 2003.

[67] E. E. Lewis, *Introduction to Reliability Engineering.* John Wiley and Sons, Inc., 1996.

[68] S. Lee and V. K. Madisetti, "Parameter optimization of robust low-bit-rate video coders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 849–855, Sep. 1999.

[69] H. C. Kim, H. Ogunleye, O. Guitart, and E. J. Delp, "The Watermark Evaluation Testbed (WET)," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, Proceedings of SPIE Electronic Imaging, (San Jose, CA), pp. 236–247, Jan. 2004.

[70] H. R. Sheikh, A. C. Bovik, and L. Cormack, "No-reference quality assessment using natural scene statistics: JPEG2000," *IEEE Transactions on Image Processing*, vol. 14, pp. 1918–1927, November 2005.

[71] P. A. Viola, *Alignment by Maximization of Mutual Information.* PhD thesis, Massachusetts Inst. Technol., Boston, MA, 1995.

[72] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, "Multimodality image registration by maximization of mutual information," *IEEE Transactions on Medical Imaging*, vol. 16, pp. 187–198, Apr. 1997.

[73] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: A survey," *IEEE Transactions on Medical Imaging*, vol. 22, pp. 986–1004, Aug. 2003.

[74] P. Viola and W. M. W. III, "Alignment by maximization of mutual information," *International Journal of Computer Vision*, vol. 24, no. 2, pp. 137–154, 1997.

[75] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* John Wiley & sons, 1991.

[76] S. S. D. Guo and S. Verdu, "Mutual information and minimum mean-square error in gaussian channels," *IEEE Transactions on Information Theory*, vol. 51, pp. 1261–1282, April 2005.

[77] P. Bas, "A quantization watermarking technique robust to linear and non-linear valumetric distortions using a fractal set of floating quantizers," *Information Hiding Workshop*, (Barcelonna, Spain), May 2005.

[78] C. A. Poynton, *A technical introduction to digital video.* New York, NY, USA: John Wiley & Sons, Inc., 1996.

[79] M. D. Srinath, P. K. Rajasekaran, and R. Viswanathan, *Introduction to statistical signal processing with applications.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[80] A. M. Tekalp, *Digital video processing.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[81] N. B. S. P. Salama and E. J. Delp, "Error concealment in encoded video streams," *Signal Recovery Techniques for Image and Video Compression and Transmission*, pp. 199–233, Kluwer Academic Publishers, 1998.

[82] P. Salama, N. B. Shroff, and E. J. Delp, "Error concealment in mpeg video streams over atm networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 1129–1144, June 2000.

[83] G. Wolberg, *Digital Image Warping.* Los Alamitos, CA: IEEE Computer Society Press, 1990.

[84] N. A. Dodgson, "Quadratic interpolation for image resampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, 1997.

[85] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing.* Cambridge (UK) and New York: Cambridge University Press, second ed., 1992.

[86] "http://www.gnu.org/software/gsl/."

[87] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer graphics: principles and practice (2nd ed.).* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[88] T. Fawcett, "Roc graphs: Notes and practical considerations for researchers," 2004.

[89] "http://www.petitcolas.net/fabien/watermarking/stirmark/."

[90] "http://poseidon.csd.auth.gr/optimark/."

[91] *Deliverables D21: Watermarking applications and requirements for benchmarking.* The Certimark Consortium, Oct. 2000. Available from http://www.certimark.org.

[92] M. Barni, C. I. Podilchuk, F. Bartolini, and E. J. Delp, "Watermark embedding: Hiding a signal within a cover image," *IEEE Communications Magazine*, vol. 39, pp. 102–108, Aug. 2001.

[93] H. S. Malvar and D. A. F. Florencio, "Improved spread spectrum: A new modulation technique for robust watermarking," *IEEE Transactions on Signal Processing*, vol. 51, pp. 898–905, Apr. 2003.

[94] A. Piva, M. Barni, F. Bartolini, and V. Cappellini, "DCT-based watermark recovering without restoring to the uncorrupted original image," *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, pp. 520–523, 1997.

[95] "http://www.ijg.org."

[96] "http://www.m-w.com."

[97] "http://www.spec.org."

[98] "http://www.eembc.org."

[99] "http://stirmark.kaist.ac.kr."

[100] "http://www.certimark.org."

[101] "http://www.mysql.com."

[102] "http://www.gimp.org."

[103] D. Taubman and M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice.* Kluwer Academic Publishers, 2002.

[104] R. Dugad, K. Ratakonda, and N. Ahuja, "A new wavelet-based scheme for watermarking images," *Proceedings of the IEEE International Conference on Image Processing*, (Chicago, IL), Oct. 1998.

[105] B. Chen and G. W. Wornell, "Quantization index modulation methods for digital watermarking and information embedding of multimedia," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 27, pp. 7–33, Feb. 2001.

[106] B. Chen and G. W. Wornell, "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.

[107] Y. Liu, B. Ni, X. Feng, and E. J. Delp, "Lot-based adaptive image watermarking," *Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents*, vol. 5306, (San Jose, CA), Jan. 2004.

[108] E. T. Lin and E. J. Delp, "Locktography: Technical description." Internal Purdue memorandum.

[109] "http://www.petitcolas.net/fabien/watermarking/stirmark31/."

[110] J. J. Hernandez, F. Perez-Gonzalez, J. M. Rodriguez, and G. Nieto, "Performance analysis of a 2-d multipulse amplitude modulation scheme for data hiding and watermarking still images," *IEEE Journal of Selected Areas in Communication*, vol. 16, no. 4, pp. 510–524, 1998.

[111] J. R. Hernandez and F. Perez-Gonzalez, "Statistical analysis of watermarking schemes for copyright protection of images," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1142–1166, 1999.

[112] M. L. Miller and J. A. Bloom, "Computing the probability of false watermark detection," *Proceedings of the Third International Workshop on Information Hiding*, pp. 146–158, 1999.

VITA

VITA

Hyung Cook Kim received his B.S. and M.S. degrees in electrical engineering from Korea Advanced Institute of Science and Technology, in 1995 and 1997, respectively. From 1997 to 1999, he fulfilled his obligatory military service in the US-ROK Combined Forces Command. In 1999, he started his studies toward the Ph.D. degree in the School of Electrical and Computer Engineering at Purdue University, West Lafayette, Indiana. He joined the Video and Image Processing Laboratory in 2000. In the spring of 2002, he served as a TA for the undergraduate digital signal processing course. In the summer of 2001 and 2002, he worked as a summer intern for the DSPS R&D center at Texas Instruments, Dallas, Texas. His research interests are in the areas of video and image coding, video and image processing, watermarking algorithms, and watermark evaluation.