ABSTRACT

STUDY OF STEGANALYSIS METHODS

by
Wen Chen

Steganography is the art and science of hiding the presence of communication by embedding secret message into digital images. The steganographic techniques can be applied in many areas. On one hand, people can benefit from their application. For example, they can be used for copyright protection. On the other hand, these techniques may be used by criminals or terrorists for malicious purposes. For example, terrorists may use the techniques to transmit the secret plan of terror attacks. Law enforcement and security authorities have concerns about such activities. It is necessary to devise the methods to detect steganographic systems. The detection of steganography belongs to the field of steganalysis.

Steganalysis is the art and science of detecting embedded message based on visual inspection, statistical analysis or other methods. Steganalysis detection methods can be classified into two categories: specific and general detection. The specific detection methods deal with the targeted steganographic systems, while the general detection methods provide detection regardless of what the steganographic systems are.

The typical detection methods are reviewed and studied in this thesis. In addition, a general detection based on artificial neural network is proposed. The performance of this detection method on detecting a generic quantization index modulation (QIM) technique is presented.

STUDY OF STEGANALYSIS METHODS

by
Wen Chen

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

January 2005

APPROVAL PAGE

STUDY OF STEGANALYSIS METHODS

Wen Chen

Dr. Yun-Qing Shi, Thesis Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Ali N. Akansu, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Constantine N. Manikopoulos, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Wen Chen

Degree: Master of Science

Date: January 2005

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2005

- Master of Science in Computer Control and Automation,
  Nanyang Technological University of Singapore, Republic of Singapore, 2001

- Bachelor of Science in Electrical Engineering,
  Xiamen University, Xiamen, P. R. China, 1991

Major: Electrical Engineering

To my beloved family

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Yun Q. Shi, who not only served as my research advisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance.

Special thanks are given to Dr. Ali N. Akansu and Dr. Constantine N. Manikopoulos for actively participating in my committee.

I am also grateful to Dekun Zhou and Chunhua Chen for their cooperation in the research.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Objective

With the wide use and abundance of steganography tools on the internet, criminals and terrorists may use them for the trafficking of the illicit material hidden in the digital images, audio and other files. Law enforcement and security authorities have concerns about these activities. It is important to find methods to detect the steganographic techniques. Detection of steganography belongs to the field of steganalysis.

The objective of this thesis is to survey and study the detection methods that can be used to defeat the steganography. Several different detection approaches will be described. The thesis is organized as follows.

In this chapter, Chapter 1 gives a brief introduction of the general concepts about steganography and steganalysis.

Chapter 2 describes the specific detection methods which include visual detection, detection based on histogram analysis, statistics methods using spatial correlations in images and the detection of the embedding techniques in JPEG images.

Chapter 3 discusses two general steganalysis detection methods. The first detection technique is based on image quality metrics. The second method has been developed based on wavelet-based statistics. A general detection method which combines the wavelet-based statistics with artificial neural network is proposed.

In the last chapter, we summarize the steganalysis methods and point out the area in the future research.

1.2 Overview of Steganography

1.2.1 General Concepts

Steganography literally means hidden writing. It refers to the art of "invisible" communication. Aimed at secure and secret information transmission, steganographic techniques strive to hide the very presence of the message from the third party. The framework usually adopted for steganography is that of *the prisoner's problem* in which two prisoners (Alice and Bob) wish to communicate in order to hatch an escape plan without the knowledge of a third party, Wendy the warden. A general model of steganography is illustrated in Figure 1.1.

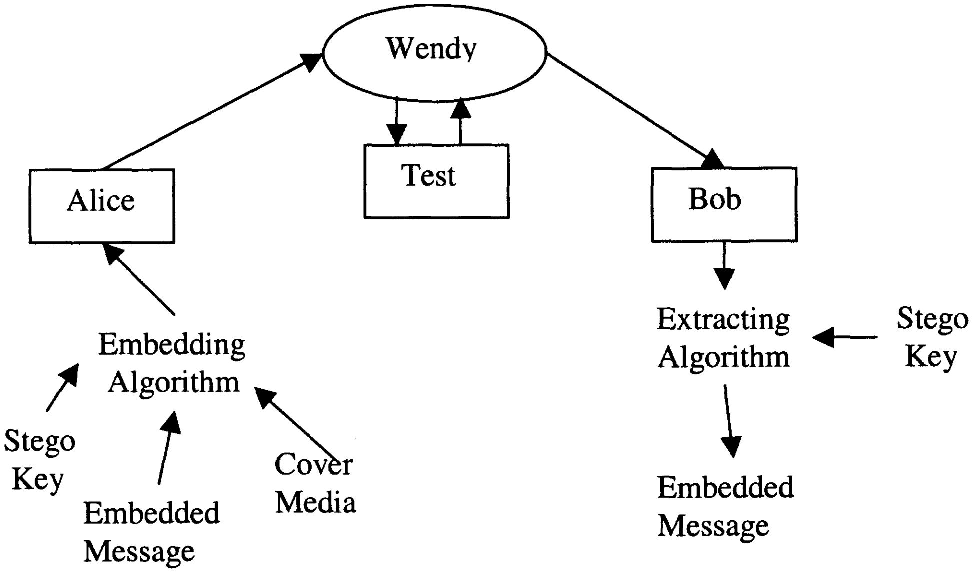


Figure 1.1 Framework of steganography.

In the figure shown above, Alice wishes to pass a secret message to Bob without alerting Wendy. To achieve this, Alice chooses an innocent looking cover medium and then embeds the message into it. The embedded process depends on a stego key which is additional secret information such as a password. The possible cover media may be audio, image, video or anything that can be represented as a bit stream. Digital images are commonly used multimedia in the Internet. They are excellent carriers for hidden information. The embedding process of image steganography is therefore may be defined as follows:

$$\text{cover image} + \text{embedded message} + \text{stego key} = \text{stego image} \qquad (1.1)$$

The system is said to be secure if Wendy can not distinguish in any sense between a cover image and a stego image.

1.2.2 Steganographic Methods

The steganographic algorithms can be categorized into two main classes: those in the image domain and those in the transform domain.

One of the populous steganographic techniques in the image domain is the least significant bit (LSB) embedding. LSB embedding hides the information by toggling the least significant bits of the pixels or the color indices to palette images. The message can be sequentially embedded or randomly spread over the image.

The transform domain steganographic methods involve the manipulation of the image transform such as discrete cosine transform (DCT). JPEG images use DCT to achieve compression. A JPEG encoder partitions an image into 8 x 8 blocks. Each block is converted to frequency coefficients using two-dimensional DCT. The DCT coefficients are quantized into integers, scanned as a one-dimensional symbol stream and then coded as a bit stream.

Several steganographic approaches in the DCT domain exist. There are three typical embedding algorithms: LSB embedding, spread spectrum and quantization index modulation.

LSB embedding method embeds message bits by replacing the least significant bits of quantized DCT coefficients. The message can be embedded sequentially in a block of DCT coefficients in the style of Jsteg [7], or scattered among the coefficients in the style of OutGuess [22]. Instead of LSB toggling, the F5 algorithm [2] uses matrix encoding and subtraction to embed a message in the DCT coefficients.

Spread spectrum algorithm hides information by modifying perceptually significant quantized DCT coefficients.

After the discrete cosine transform, the DCT coefficients are quantized into integers. This is done by dividing each coefficient by a constant for that coefficient, and then rounding to the nearest integer. Quantization index modulation hides message by manipulating the rounding choices either up or down in the DCT coefficients during quantization process.

The advantage of embedding in the DCT domain is that the distortion due to the embedding operation is spread to a group of pixels, making it even more difficult to be visibly detected.

1.3 Detection of Hidden Information

The purpose of steganography is to transmit hidden information embedded in a cover image in such a stealth way that no third party can discover the very presence of embedded message. Although creative approaches have been devised in the data embedding process to make the detection difficult, it is still possible to detect the existence of the hidden data. The art and science of detecting the existence of embedded message is called steganalysis which is classified into two categories: specific and general detection.

Specific detection methods deal with targeted steganographic systems. Each detection method will be statistic (a function of the investigated image) designed to discriminate between the two cases: an image with and without hidden message. For example, the statistic may be the *p*-value of an observation which is the probability that the image is embedded with hidden information. If a high *p*-value is obtained, the image is claimed to have a message embedded. The discriminating statistic may also be the estimate of the embedded message length. If the estimate is greater than a threshold, the image is classified as containing an embedded message.

The general detection methods can be applied to any steganographic system after proper training on a set of images. General detection methods involve the extraction of a feature vector and the classification which employs some powerful classifiers such as support vector machines or neural networks.

CHAPTER 2

SPECIFIC STEGANALYSIS METHODS

2.1 Introduction

As indicated in Chapter 1, steganography is the art of embedding confidential information into a cover media. Its purpose is to avoid the detection of the very presence of a secret message. But hiding information in electronic media may introduce statistically detectable artifacts that reveal the embedding of a message or the embedded message itself.

There is a significant body of work in the area of specific detection. The focus of this chapter is on the detection of the LSB embedding techniques. LSB embedding techniques hide message bits by replacing the least significant bits of either pixel values, or color indices to the palette, or quantized DCT coefficients.

The specific detection consists of subjective and statistical methods. The subjective methods make use of human eyes to look for suspicious artifacts in the image. The statistical methods perform mathematical analysis of the images to find the discrepancy between the original and stego images.

The first statistical method of the specific detection is based on statistical analysis of the sample values in pairs. It becomes known as the chi-square attack which provides very reliable detection of the message sequentially embedded in the image. Although this method is quite successful for sequential embedding or large message, it fails somewhat for randomly embedded messages.

RS Analysis can provide reliable and accurate detection of randomly embedded message. An image is partitioned into "regular" or "singular" groups depending on the noisiness of a given group. The proportion of "regular" and "singular" groups forms the curves that are quadratic in the amount of message embedded. If some assumptions are satisfied, the message length can be accurately estimated.

Another effective method is Pairs Analysis. An image is first split into "color cuts" for color pairs. For a given color pair, this involves generation of a binary sequence, associating a "0" with the first color and a "1" with the second color. After Concatenating the color cuts into a single stream and measuring the homogeneity of the LSBs, the attacker can collect enough information to deduce the amount of the embedded data in an image under some assumptions.

To launch an attack on the LSB embedding in the transform domain, the histogram of the quantized DCT coefficients are used to compute the statistics that predictably changes with the embedded message length.

Several specific methods to detect LSB embedding are described in the rest of this chapter.

2.2 Subjective Methods

2.2.1 Ezstego Algorithm

The Ezstego embedding algorithm was created by Romana Machado [26]. It sequentially embeds the hidden message in the palette indices of GIF images. The GIF image is an indexed image which consists of an array and a color palette. The pixel values in the array are direct indices into a palette. The color palette contains up to 256 different colors with values in the range [0, 1]. Each row of the palette specifies the red, green and blue components of a single color. An indexed image uses direct mapping of pixel values to the color in the palette. The color of each image pixel is determined by using the corresponding pixel value as an index into the palette. Figure 2.1 [10] illustrates the structure of an indexed image. In the figure, the pixel value 5 points to the fifth row of the palette which indicates the combination of color component for this pixel (R=0.2902, G=0.0627, B=0.0627).

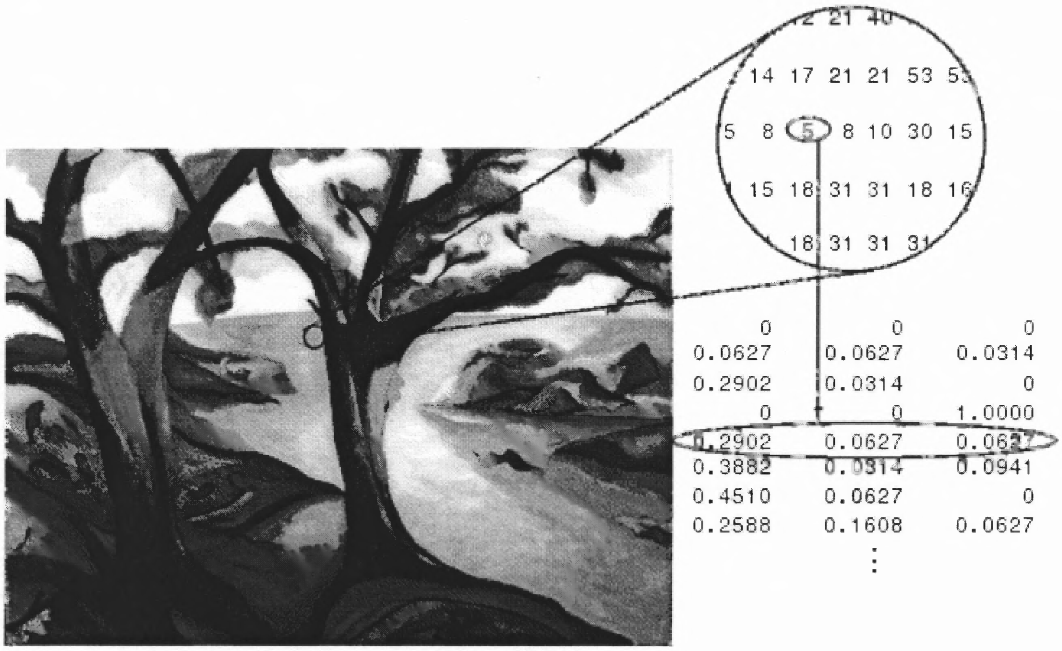


Image Courtesy of Susan Cohen

Figure 2.1 GIF image structure.

The first step of the algorithm is to create a sorted copy of the palette in which the difference between two adjacent colors is minimized. During the embedding, the message bits are embedded as the LSBs of color indices to the sorted palette. The embedding algorithm matches the hidden message bit with the LSB of an index to be embedded, and replaces the color by its neighboring color in the sorted palette if necessary.

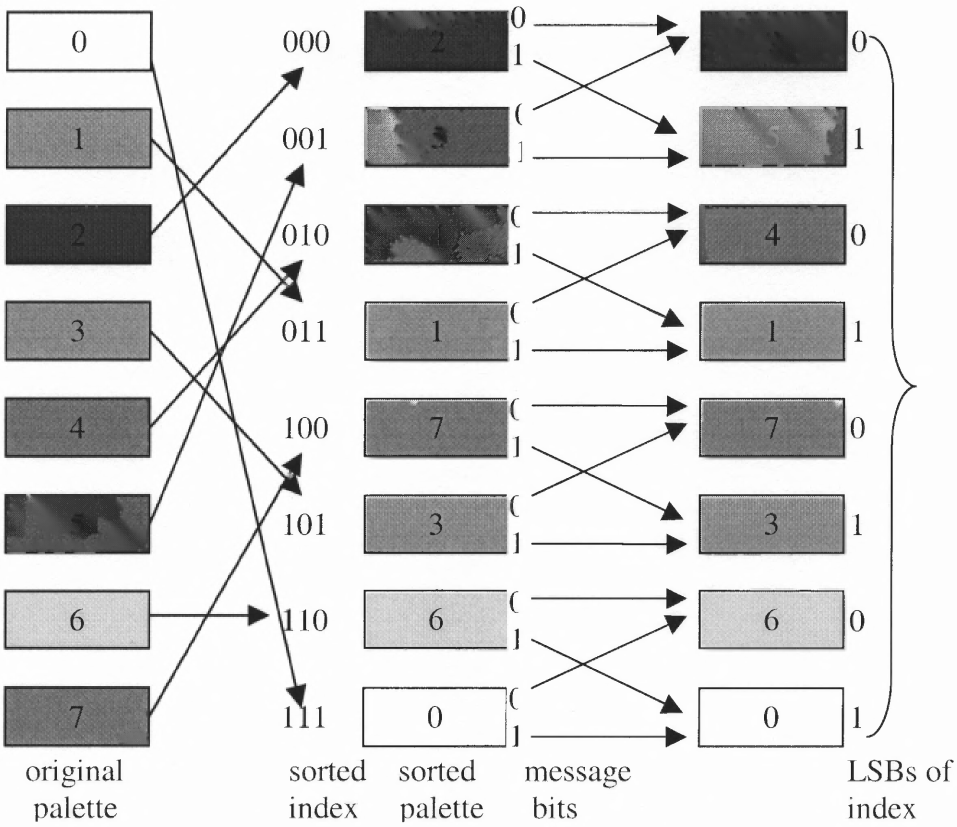


Figure 2.2 Ezstego embedding.

The embedding mechanics is demonstrated in Figure 2.2 with a reduced palette having eight different colors. The number 0 to 7 inside the shaded rectangular represents the pixel value in the GIF file. Assume that we want to embed a "0" in the pixel of value 5 which corresponds to the index 1 (001) in the sorted palette. The index 1 becomes index 0 with the least significant bit of index 1 replaced with the message bit "0", i.e. $00\underline{1} \rightarrow 00\underline{0}$. The color of pixel 5 is therefore replaced by its neighbor color of index 0 in the sorted palette. Because the difference between these two colors is very small, the change is almost imperceptible for human eyes.

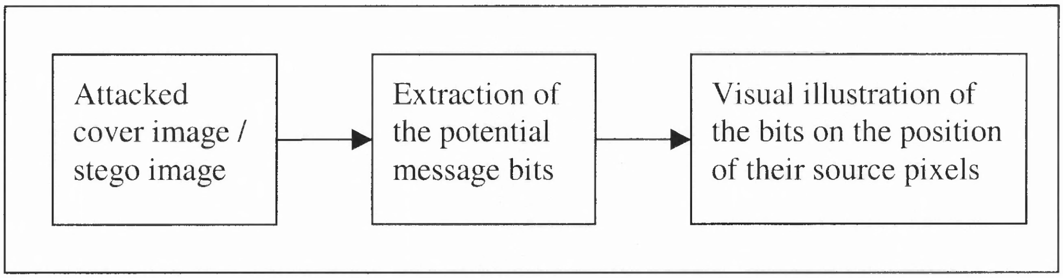
The extraction of the message bits is very easy. Before extracting operation, a sorted copy of palette is first created, then the message values can extracted from the

LSBs of the sorted indices.

2.2.2 Visual Attack

The visual attacks [1] detect the steganography by making use of the ability of human eyes to inspect the images for the corruption caused by the embedding.

The idea of visual attacks is to remove all parts of the image covering the message. The human eye can now identify the potential hidden message using the visual inspection after pre-filtering the stego images. The filtering process has the following structure:


Attacked cover image / stego image
Extraction of the potential message bits
Visual illustration of the bits on the position of their source pixels

The visual attacks work well for the image with connected areas of uniform color or areas with the color saturated at either 0 or 255. This approach is especially suited to palette images with LSB embedding in indices to the palette.

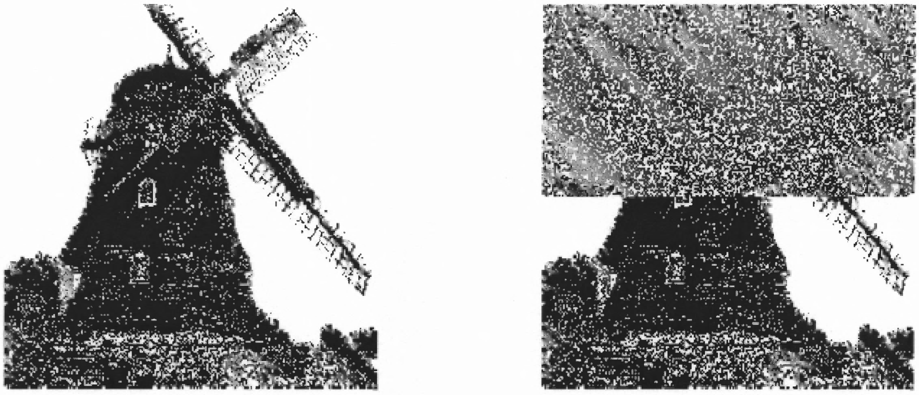


Figure 2.3 The original image and 50% of the image embedded with Ezstego.

Figure 2.3 shows that the top half of the lowpass-filtered stego image is corrupted after embedding. There is a clear dividing line between the embedded and the unchanged part.

Visual attacks are very simple, but they are hard to implement automatically and the reliability is highly questionable. The more powerful and reliable detection schemes are statistical methods that will be presented in the following.

2.3 Chi-Square Attack

Pfitzman and Westfeld [1] introduced a powerful statistical attack based on histogram analysis of Pairs of Values (PoVs) that are swapped during message embedding process. The PoVs can be formed by pixel values, quantized DCT coefficients, or palette indices that differ in the least significant bit.

If the message bits are equally distributed, the occurrences of both values in each pair become equal. In other words, the distribution of occurrences of the two values from each pair will have a tendency to become equal after message embedding (this depends on the message length). The histograms in Figures 2.4 & 2.5 illustrate the tendency. Suppose there are six pairs of colors in the histogram of colors.

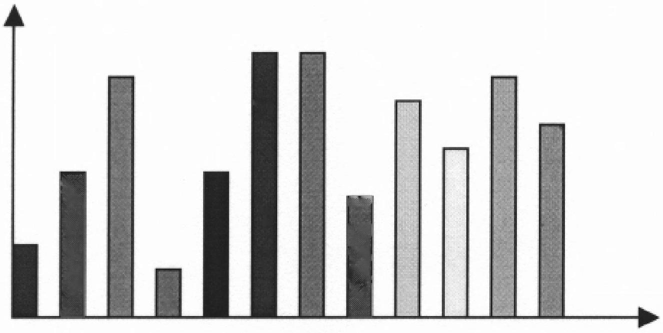


Figure 2.4 The color histogram before embedding.

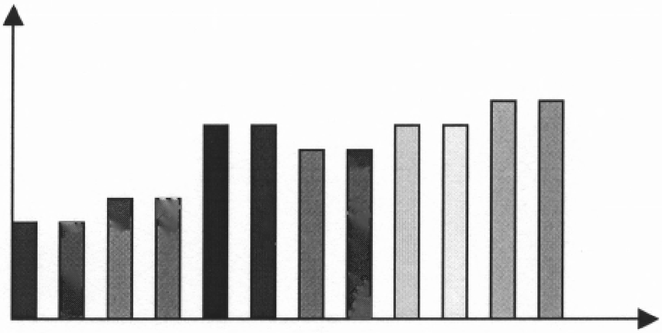


Figure 2.5 The color histogram after embedding.

The idea of this attack is to test for the statistical significance of the fact that the occurrences of both values in each pair are the same. That means the comparison of the theoretically expected frequency distribution and some sample distribution observed in the attacked image. The Chi-square can be used to determine the goodness of fit of the expected distribution to the sample distribution.

A critical point for the design of this test is to derive the theoretically expected frequency distribution. In the original, the theoretically expected frequency is the arithmetic mean of the two frequencies in a PoV. Since swapping one value into another does not change the sum of occurrences of both values in each pair, the arithmetic mean of the two frequencies for each pair is the same in both the original and stego image. This fact allows us to obtain the expected frequency from the stego image.

Once the observed sample distribution and the theoretically expected frequency distribution are determined, the Chi-square test can be used to determine the degree of similarity between them. The Chi-square attack works as follows:

1. Suppose that there are k categories and that we have a random sample of observation. Each observation must fall within only one category. For example, for a palette image there are at most 256 colors c_i in the palette, which means at most 128 PoVs and $k = 128$.

2. the theoretically expected frequency in *ith* category, $i = 1, 2, \ldots, k$ after embedding an equally distributed message bits is defined as

$$n_i \quad \frac{\text{number of indices in the pair } \{c_{2i}, c_{2i+1}\}}{2} \tag{2.1}$$

3. the actual frequency of occurrence in the sample is

$$n_i' = \text{number of indices to } c_{2i} \tag{2.2}$$

4. the Chi-square statistic is calculated as

$$\chi^2_{k-1} = \sum_{i=1}^{k} \frac{(n_i - n_i')^2}{n_i'} \tag{2.3}$$

with k-1 degrees of freedom.

5. p is the probability of the above statistic under the condition that the distributions of n_i' and n_i are equal. It is given by integration of the density function:

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma(\frac{k-1}{2})} \int_0^{\chi^2_{k-1}} e^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} dx \tag{2.4}$$

If the distribution of n_i tends to be equal to that of n_i', χ^2_{k-1} will approach to zero, accordingly p value will be close to one.

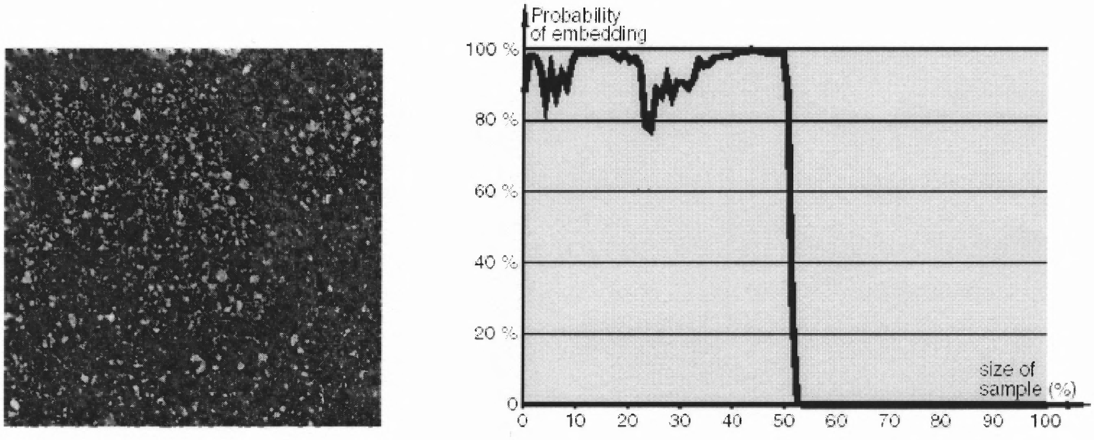


Figure 2.6 Probability of embedding with Ezstego in the flooring tile image.

Figure 2.6 presents the p–value of the Chi-square test as a function of an increasing sample. The flooring tile is embedded sequentially with a message of 3600 bytes. The image is scanned in the order in which the message has been embedded and the p value is evaluated as a function of an increasing sample which consists of the set of all already visited pixels. Initially 1% of the pixels are scanned starting from the upper left corner. For this sample the p value is 0.8826. The p value increases to 0.9808 when an additional 1% of the pixel is included in the sample. As long as the sample pixels are all embedded with message bits, the p value does not drop below 0.77. The p value drops to zero when the end of the embedded message is reached. It remains zero until the lower right corner of the image.

The Chi-square test works well for sequential embedding. If the embedded message is randomly spread over the cover image, the test is less reliable unless the embedded bits are hidden in the majority of the pixels. On the other hand, any detection technique based on the analysis of the histogram will be easy to circumvent. For an embedding technique that will preserve the original counts of sample in their PoVs, it is difficult to detect the hidden massage by this attack.

2.4 RS Analysis

To reliably and accurately detect the hidden message that is randomly scattered through the image, Fridrich et al. [13] introduced a powerful steganalysis method that utilizes the spatial correlation in the stego image. The basic idea is to discover and quantify the weak relationship between the LSB plane and the image itself. From the relationship a measure that is the function of the embedded message can be determined. An effective method can be devised by studying how this measure changes with the hidden message.

2.4.1 Terminology

Given an *M*-by-*N* image whose pixel values belong to the set *P*, the image is first partitioned into groups of n adjacent pixels $(x_1,\ldots,x_n)$ along the rows or columns. To capture the spatial correlation, a discrimination function *f* is defined as the mean absolute value of the differences between adjacent pixels.

$$f(x_1,\ldots,x_n) = \frac{1}{n\text{-}1}\sum_{i=1}^{n}\left|x_{i+1} - x_i\right| \tag{2.5}$$

This measures the smoothness or regularity of the pixel group $G = (x_1,\ldots,x_n)$. The noisier the pixel group is, the larger the value of the discrimination function *f* becomes.

The LSB embedding increases the noisiness in the image, and thus the value of the discrimination function *f* will increase after flipping the LSBs of a fixed set of pixels within each group. The LSB embedding can be described using standard flipping function F_1 and dual flipping function F_{-1} as follows:

$$F_1:\ 0 \leftrightarrow 1, 2 \leftrightarrow 3, \ldots, 254 \leftrightarrow 255$$

$$F_{-1}:\ \text{-}1 \leftrightarrow 0, 1 \leftrightarrow 2, \ldots, 255 \leftrightarrow 256$$

$$F_0:\ F_0(x) = x \quad \forall x \in P$$

The pixel group *G* can be classified into three different types: *R*, *S*, and *U* depending on how the flipping changes the value of the discrimination function:

$$\text{Regular groups } R \leftrightarrow f(F(G)) > f(G)$$

$$\text{Singular groups } S \leftrightarrow f(F(G)) < f(G)$$

$$\text{Unchanged groups } U \leftrightarrow f(F(G)) = f(G)$$

Here, $F(G)$ applies the flipping function F to each component of the group $G = (x_1, \ldots, x_n)$. For a group to be regular, the pixel noise within this group should be increased after LSB flipping. Similarly, for a singular group, the pixel noise becomes decreased after the flipping operation.

In general, the different flipping is applied to the different pixel in the group G. The pattern of pixels to flip is called "mask" which consists of zeros and ones. The mask has the same size as the group to flip, determining how pixels in each group are flipped. Thus we define the flipped group $F(G)$ as $(F_{M(1)}(x_1),\ F_{M(2)}(x_2),\ \ldots,\ F_{M(n)}(x_n))$, where $M(i),\ i = 1,\ 2,\ \ldots,\ n$ is the element of mask M and takes on the values -1, 0, and 1. For example, if $M = (0, 1, 1, 0)$, then $F(G) = (F_0(x_1),\ F_1(x_2),\ F_1(x_3),\ F_0(x_4))$.

2.4.2 The Principle of RS Analysis

Since the LSB flipping simulates the act of adding pixel noise, it more frequently results in an increase in the value of the discrimination function f rather than a decrease. Thus the total number of regular groups will be larger than that of singular groups. Let R_M and S_M be the relative number of regular groups and singular groups for a non-negative mask M, respectively. The following zero-message hypothesis is true for the typical cover images, that is

$$R_M \cong R_{-M} \tag{2.6}$$

and

$$S_M \cong S_{-M} \tag{2.7}$$

which mean that the value of R_M is approximately equal to that of R_{-M} if no hidden message exists. The same should be true for the relationship between S_{+M} and S_{-M}.

However, the above assumption does not hold if the LSB plane is randomized. R_M and S_M have a tendency to close to each other as the embedded message length increases. The difference between R_M and S_M will approach to zero if the LSBs of 50% pixels are flipped. In such case R_M and S_M have the following relationship:

$$R_M \cong S_M \,. \tag{2.8}$$

Surprisingly randomization of the LSB plane forces the difference between R_{-M} and S_{-M} to increase with the embedded message length.

The experiment [13] shows that the R_{-M} and S_{-M} have approximately linear relationship with respect to the number of pixels with flipped LSBs and thus are defined as straight lines, while R_M and S_M curves can be approximated by the parabolas. Figure 2.7 demonstrates R_M, S_M, R_{-M}, and S_{-M} as functions of the number of pixels with flipped LSBs. The graph is called RS diagram in which the x-axis represents the percentage of pixels with flipped LSBs, the y-axis is the relative number of regular and singular groups with masks M and –M, M = [0,1,1,0].

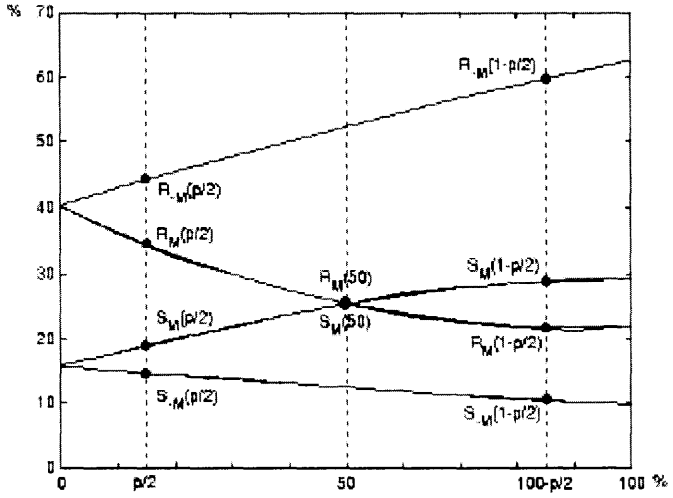


Figure 2.7 RS-diagram.

Suppose the hidden message length in a stego image is p (in percent of pixels). The points $R_M(p/2)$ and $S_M(p/2)$ correspond to the number of R and S groups for the non-negative mask M. Similarly we have the points $R_{-M}(p/2)$ and $S_{-M}(p/2)$ for the mask $-M$. The reason that the x-coordinate of these four points is only one half of the embedded message length p is that on average the probability that a pixel will be flipped is 0.5 if the embedded message is assumed to be a random bit-stream.

The other four points $R_M(1\text{-}p/2)$, $S_M(1\text{-}p/2)$, $R_{-M}(1\text{-}p/2)$, and $S_{-M}(1\text{-}p/2)$ are obtained by flipping the LSBs of all pixels in the stego image and applying the flipping operation. The middle points $R_M(1/2)$ and $S_M(1/2)$ will be obtained by randomizing the LSB plane of the stego image. The points $R_{-M}(p/2)$, $R_{-M}(1\text{-}p/2)$ and $S_{-M}(p/2)$, $S_{-M}(1\text{-}p/2)$ define two straight lines. The points $R_M(p/2)$, $R_M(1/2)$, $R_M(1\text{-}P/2)$ and $S_M(p/2)$, $S_M(1/2)$, $S_M(1\text{-}P/2)$ determine two parabolas.

It is possible to find a formula for the calculation of the embedded message length, if the following two assumptions hold.

1. The curves R_M and R_{-M} intersect at the same x coordinate as the curves S_M and S_{-M}.
2. The point of intersection of curves R_M and S_M has the x coordinate to be equal to 50%, that is, $R_M(1/2) = S_M(1/2)$.

To derive the formula the x-axis is first be rescaled so that $p/2$ becomes 0 and $100\text{-}p/2$ becomes 1. The x-coordinate of the intersection point is the root of the following quadratic equation:

$$2(d_1 + d_0)x^2 + (d_{-0} - d_{-1} - d_1 - 3d_0)x + d_0 - d_{-0} = 0 \tag{2.9}$$

where $d_0 = R_M(p/2) - S_M(p/2)$, $d_1 = R_M(1\text{-}p/2) - S_M(1\text{-}p/2)$, $d_{-0} = R_{-M}(p/2) - S_{-M}(p/2)$, and $d_{-1} = R_{-M}(1\text{-}p/2) - S_{-M}(1\text{-}p/2)$. The secret message length p can be obtained by

$$p = x/(x - 1/2) \tag{2.10}$$

where x is the root taking on the small absolute value.

2.4.3 The Accuracy of RS

Under certain assumptions the amount of embedded message could be accurately determined if the numbers of regular and singular groups are given. There are several factors that influence the accuracy of this detection method: the initial bias, the selection of mask, the noise level of the cover image, and the placement of message bits in the image.

The initial bias is the estimate of hidden message length when no data is actually hidden. Theoretically the embedded message length should be zero for original cover images. But the RS analysis may indicate a small non-zero initial bias which could be either positive or negative and puts a limit on the theoretic accuracy of RS analysis. Smaller images have smaller number of regular and singular groups and thus tend to have higher variation in the initial bias. On the other hand, the bias is typically small for JPEG images, uncompressed images obtained by a digital camera, for sans of photographs, and for images processed with typical image processing filters. Generally color images have larger variations than grayscale images in the initial bias.

The RS statistic depends on how the image is partitioned into groups of a fixed shape. The mask determines the pattern of group pixels to flip and thus has significant effect on the accuracy of RS steganalysis. In [3], various masks have been tried to investigate their influence on the reliability of RS steganalysis for totally 10000 JPEG images of moderate quality, each with 5% of the pixels used for LSB embedding. These masks include the flat masks [0,1], [0,1,0],[0,1,1,0],[0,1,1,1,0], [0,1,0,1,0] and the square masks:

$$M_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, M_{3a} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, M_{3b} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, M_{4a} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, M_{4a} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

It's found that there are some small but significant differences between the results with the various masks. The masks [0, 1, 0] and M_{3a} perform the best with the square mask slightly better than the flat mask in some cases. M_2, [0, 1, 1, 0] and [0, 1, 0, 1, 0] are the next-best performers. [0, 1, 1, 1, 0], M_{3b}, M_{4a} and M_{4b} perform the worst.

For very noisy images, the difference between the number of regular and singular pixels in the cover image is small. Consequently, the lines in the RS diagram intersect at a small angle and the accuracy of RS decreases. Finally, the RS steganalysis is more accurate for messages that randomly spread over the stego image than for images with sequentially embedded message.

The RS analysis provides very reliable detection for the secret message randomly embedded in the stego images only under certain assumptions. If these assumptions do not hold, this detection will fail.

2.5 Pairs Analysis

Pairs analysis was proposed by Fridrich et al [17]. This approach is well suited for the embedding archetype that randomly embeds messages in LSBs of indices to palette colors of palette image. The EzStego algorithm is an example of such embedding archetype. The original EzStego algorithm which was introduced in Subsection 2.2.1 is designed for sequential message embedding. To demonstrate the superior performance of

Pairs analysis, the EzStego method is modified so that the message bits are embedded in a random walk.

Before the message bits are embedded, EzStego first finds an optimized palette order which minimize color differences between consecutive colors in the palette entry. For the palette colors $c_0, c_1, \cdots, c_{p-1}, P \le 256$, a sorted palette $c_{\pi(0)}, c_{\pi(1)}, \cdots, c_{\pi(p-1)}$ is found so that $\sum_{i=0}^{p-1} \left| c_{\pi(i)} - c_{\pi(i+1)} \right|$ is minimal ($c_{\pi(p)} = c_{\pi(0)}$). Here π represents the sorting permutation determined by EzStego. This sorted palette forms a set of embedding pairs *E*

$$E = \{(c_{\pi(0)},\ c_{\pi(1)}),\ (c_{\pi(2)},\ c_{\pi(3)}),\ \ldots,\ (c_{\pi(p-2)},\ c_{\pi(p-1)})\} \tag{2.11}$$

and the shifted pairs *E'*

$$E' = \{(c_{\pi(1)},\ c_{\pi(2)}),\ (c_{\pi(3)},\ c_{\pi(4)}),\ \ldots,\ (c_{\pi(p-1)},\ c_{\pi(0)})\} \tag{2.12}$$

Pairs Analysis first splits an image into a set of *color cuts* by scanning the whole image by rows and/or columns. Each color pair from *E* or *E'* corresponds to a *color cut* which is a binary vector. For example, to obtain the *color cut* for the *embedding pair* (c_1, c_2), the whole image is scanned and only pixels with colors falling into (c_1, c_2) are selected to generate a sequence of colors. The color sequence is then converted to a binary sequence by associating "0" with the first color c_1 and "1" with the second color c_2.

The *color cuts* are concatenated into a single stream. Let $Z(c_{2k}, c_{2k+1})$ represent the color cut for the embedding pair (c_{2k}, c_{2k+1}), the concatenated *color cuts* for all pairs in E is

$$Z = Z(c_{\pi(0)}, c_{\pi(1)}) \,\&\, Z(c_{\pi(2)}, c_{\pi(3)}) \,\&\cdots\&\, Z(c_{\pi(p-2)}, c_{\pi(p-1)}) \tag{2.13}$$

Similarly, for all shifted pairs *E'*

$$Z' = Z(c_{\pi(1)}, c_{\pi(2)}) \,\&\, Z(c_{\pi(3)}, c_{\pi(4)}) \,\&\cdots\&\, Z(c_{\pi(p-1)}, c_{\pi(0)}) \tag{2.14}$$

Based on the bit-streams Z and Z', a quantity called homogeneity can be defined. The homogeneity measures the structure in the bit-streams Z and Z'. Let $R(p)$ be the homogeneity of Z after flipping the LSBs of indices of p percent of pixels along a random walk ($0 \le p \le 1$).

$$R(p) = \frac{\text{the number of homogenous bit-pairs ('00','11') in } Z}{\text{the length of } Z} \tag{2.15}$$

The homogeneity of Z' denoted as $R'(p)$ is similarly defined.

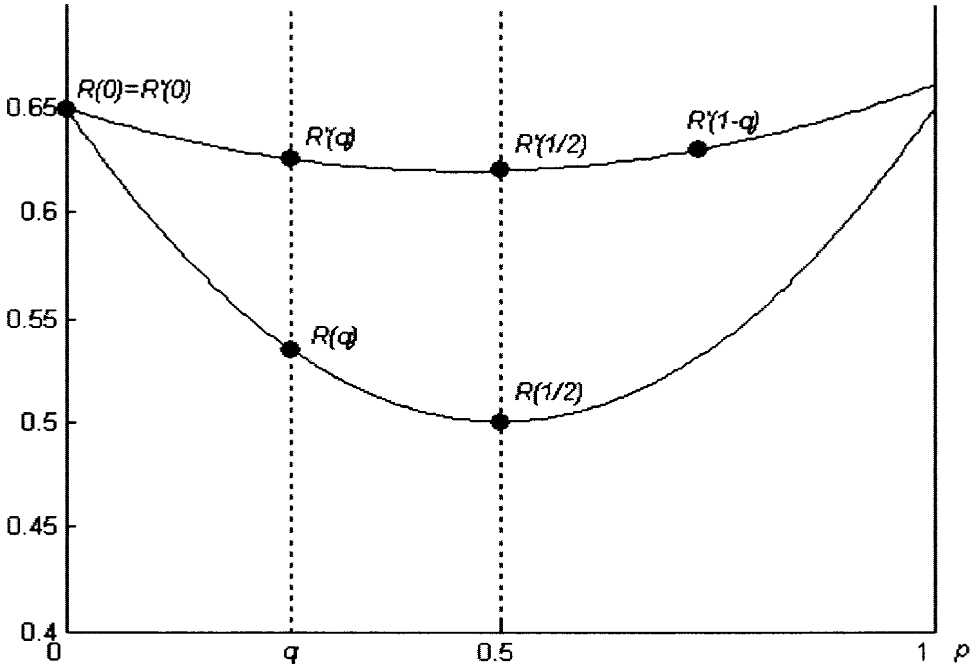


Figure 2.8 Typical Pairs diagram.

The curve $R(p)$ (See Figure 2.8) is a parabola with a minimum vertex at p=0.5 and R(0.5) is equal to 0.5. The stego image with an unknown message length q produces the homogeneity $R(q)$. Also note that $R(q)$ = $R(1\text{-}q)$. So the parabola $R(p)$ is uniquely determined by three points (0.5, R(0.5)), (q, $R(q)$) and (1-q, R(1-q)).

The curve R' (p) is well modeled using a second-polynomial through the points

(0.5, $R'(0.5)$), (q, $R'(q)$) and (1-q, $R'(1\text{-}q)$). The value $R'(q)$ can be calculated from the stego image. Unlike the case of $R(p)$, $R'(q) \neq R'(1\text{-}q)$ does not hold. The value of $R'(1\text{-}q)$ can be derived from the stego image with all color flipped. The value $R'(0.5)$ is determined based on the following theorem:

Theorem: let $Z' = \{b_i\}_{i=1}^{n}$ be the binary sequence defined above, the expected value $R'(0.5) = \sum_{k=1}^{n-1} h_k$, where h_k = the number of homogenous pairs in the sequence { $b_1 b_{1+k}$, $b_2 b_{2+k}$, $b_3 b_{3+k}$, … , $b_{n\text{-}k} b_n$ }.

Pairs Analysis can work well only under the assumption that "natural images" have equal homogeneity, that is, $R(0) = R'(0)$. The assumption implies that the color cuts of embedding pairs E and shifted pairs E' should have the same structure. Under the assumption of equal homogeneity, it can be shown that the difference of the two homogeneity measures $R(p)$ and $R'(p)$ is a function that is quadratic in the embedded message length. That's, $D(p) = R(p)\text{-}R'(p) = ap^2 + bp + c$. By eliminating the unknown parameters a, b, c, the following quadratic equation for q is obtained.

$$4D(0.5)q^2 + [D(1\text{-}q) - D(q) - 4D(0.5)]q + D(q) = 0 \qquad (2.16)$$

By solving this quadratic equation, we have two roots for the unknown q. The smaller root is the estimate of the embedding message length.

It's needed to point out that Pairs Analysis should work for grayscale images [3] although it was designed with palette images. In the case of grayscale images, the embedding pairs will become E = {(0, 1), (2, 3),… (254, 255)}, and the shifted pairs will be E' = {(255, 0), (1, 2),… (253, 254)}.

The experimental results show that the method of Pairs Analysis exhibits the reliable and accurate message detection for GIF images of natural scene but it is not as reliable for cartoons and computer-generated images (fractals). The reason that Pairs Analysis provides lower detection accuracy is mainly because the assumption of equal homogeneity does not hold in the case of artificial images. If the assumption is violated ($R(0) \neq R'(0)$), the initial bias which is the message length detected in the original cover image may not be very close to zero. Artificial images such as cartoons and fractals have a low number of unique colors leading to some singular structure in the embedding pairs or shifted pairs, which causes the assumption to become invalid.

2.6 Steganalysis of JPEG Image

2.6.1 JEPG Format

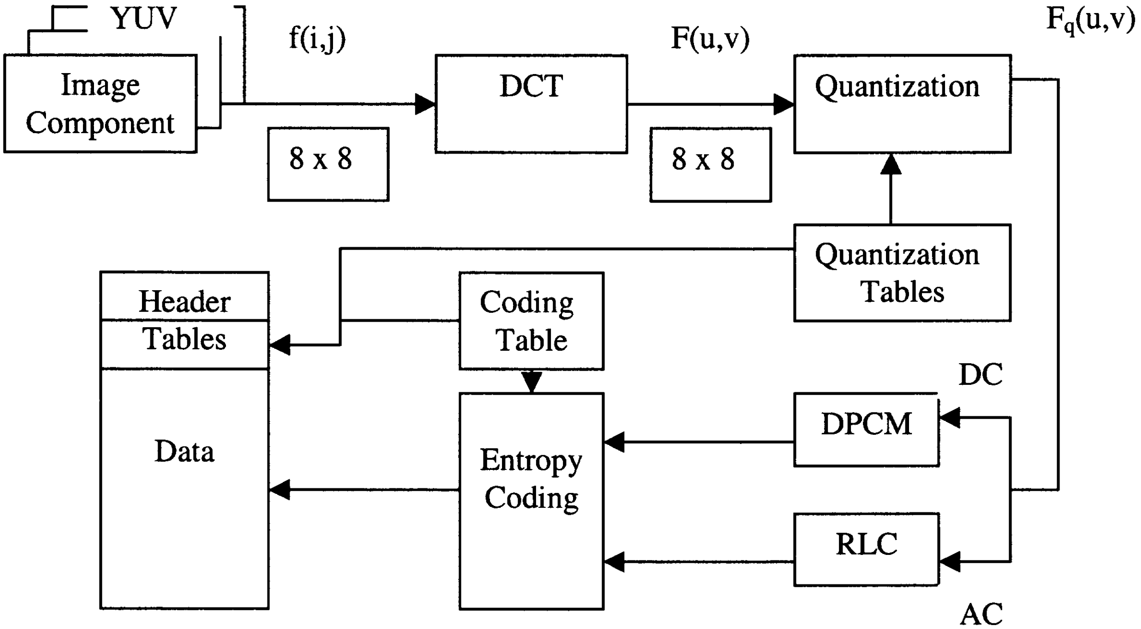


Figure 2.9 JPEG encoder.

In addition to steganographic techniques in the spatial domain, there are robust steganographic methods in the transform domain where the JPEG format is frequently adopted as the carrier medium.

Figure 2.9 shows how a JPEG image is generated. First the image is converted from RGB into a different color space called YUV. The Y component represents brightness of a pixel, and the U and V together represent hue and saturation.

Then the chroma sub-sampling is performed. This step is to reduce U and V components to half size in vertical and horizontal directions, thereby reducing the size in bytes of the whole image by a factor of two. For the rest of the compression process, Y, U and V are processed separately and in a very similar manner. The rations at which down-sampling can be done on JPEG are 4:4:4, 4:2:2 and 4:1:1.

Each component (Y, U, V) of the image is partitioned into blocks of eight by eight pixels each, then each block is converted to frequency space using a two-dimensional discrete cosine transform (DCT).

After the discrete cosine transform, the DCT coefficients are quantized. This is done by dividing each coefficient in the frequency domain by a constant for that coefficient, and then rounding to the nearest integer. As a result of this, many of the higher frequency components are rounded to zeros, while many of the rest become small positive and negative numbers.

The quantized components are placed in a "zigzag" order that groups similar frequencies together, then do run-length coding on zeros and Huffman coding on what's left.

2.6.2 F5 Embedding Algorithm

The F5 algorithm was introduced by German researchers Pfitzmann and Westfeld in 2001 [2]. It embeds message bits into non-zero AC coefficients and adopts matrix encoding to achieve the minimal number of changes in quantized coefficients during embedding process.

The matrix encoding is the core of the F5 algorithm. It is determined by the message length and the number of non-zero AC coefficients. It can be represented as the form (c, n, k). The parameter c tells how many coefficients at most will be modified during embedding, and n is the number of coefficients involved in embedding the k-bit message. In the embedding process, the message is divided into segments of k bits to embed into a group of n randomly chosen coefficients.

To demonstrate how matrix encoding works, the triple (1, 3, 2) is used. In this situation, suppose the message bits are x_1 and x_2. We want to embed x_1 and x_2 into three non-zero AC coefficients which are assumed to have corresponding least significant bits a_1, a_2 and a_3, respectively. Since at most one coefficient is allowed to be modified, the successful embedding should fall into one of the following scenarios:

$$x_1 = a_1 \oplus a_3, \quad x_2 = a_2 \oplus a_3 \quad \Rightarrow \text{nothing changed}$$

$$x_1 \neq a_1 \oplus a_3, \quad x_2 = a_2 \oplus a_3 \quad \Rightarrow \text{only } a_1 \text{ flipped}$$

$$x_1 = a_1 \oplus a_3, \quad x_2 \neq a_2 \oplus a_3 \quad \Rightarrow \text{only } a_2 \text{ flipped}$$

$$x_1 \neq a_1 \oplus a_3, \quad x_2 \neq a_2 \oplus a_3 \quad \Rightarrow \text{only } a_3 \text{ flipped}$$

Here $\oplus$ is XOR operation. If the embedding causes the coefficient to become zero, which is called *shrinkage* and occurs when the coefficient equal to one is subtracted by 1, the same message must be embedded into the next coefficient. In general, the (1, 3, 2) matrix encoding is implemented as follows:

(a) Buffer 3 non-zero AC coefficients.

(b) Hash the buffer to generate a hash value in binary equal to s_2s_1, where $s_2 = a_2 \oplus a_3$ and $s_1 = a_1 \oplus a_3$.

(c) Perform XOR of the message bit with the hash value to produce b_2b_1, where $b_2=x_2 \oplus s_2$ and $b_1=x_1 \oplus s_1$. Let d be the converted decimal value of b_2b_1.

(d) If d is zero, nothing is changed with this group of coefficients. Otherwise the absolute value of a_d has to be decreased by one.

(e) If shrinkage occurs, feed one more new coefficient into the buffer and repeat the steps (a) to (e). Otherwise continue with new coefficients until the end of the embedded message.

The simple matrix encoding (1, 2^k-1, k) is adopted in F5 algorithm, which means the k message bits are embedded into every group of 2^k-1 coefficients with at most one coefficient in this group modified.

The F5 embedding process first finds the RGB representation of the input image and converts it to YUV space, then the brightness component Y and the color components U, V are transformed into frequency coefficients in the DCT domain. These frequency components are quantized with the predefined quality factor. The estimated embedding capacity is determined by the number of non-zero AC coefficients in the brightness channel since only those coefficients are used to embed message bits by F5. From the embedding capacity, the parameter k is determined and the number of coefficients (=2^k-1) in each group is also known. Once the matrix encoding (1, 2^k-1, k) is determined, the message can be embedded in the quantized coefficients along the key-dependent random walk. The message is segmented into portions of k bits that are embedded into a group of 2^k-1 coefficients. If the k-bit message does not match the hash of that group, the absolute value of at most one of the coefficients in the group is decremented by one to get the match. If the subtraction leads to zero coefficient, which is so-called shrinkage, the same message bit should be re-embedded in the next coefficient, because only non-zero AC coefficients will be visited when extracting the message bits at the receiving end.

2.6.3 Attack on the F5 Algorithm

If a distinguished statistics which predictably changes with the embedded message length can be identified, it is possible to successfully launch an attack on F5 [14]. In fact, the F5 algorithm manipulates the quantized coefficients when the hash of that group does not match the message bits, thus the histogram values of DCT coefficients are modified. For example, if the shrinkage occurs, the number of zero AC coefficients will increase and the number of remaining non-zero coefficients decreases with embedding. The changes in the histogram of DCT coefficients may be utilized to detect the presence of hidden message. Suppose that

(1) $h(d)$: the total number of AC DCT coefficients in the cover image with absolute value equal to $d, d = 0,\ 1, 2, \ldots$.

(2) $h_{kl}(d)$: the total number of AC coefficients of frequency (k, l) $[1 \leq k, l \leq 8]$ with absolute value equal to d.

(3) $H(d)$ & $H_{kl}(d)$ are the corresponding histogram values in the stego image.

If there are totally n non-zero AC coefficients to be modified during the embedding process, the number of relative modification of DCT coefficients would be $\beta = n/P$, where $P = h(1) + h(2) + \ldots$. The expected histogram values of the stego image are

$$H_{kl}(d) = \begin{cases} (1-\beta)h_{kl}(d) + \beta h_{kl}(d+1), & d>0 \\ h_{kl}(0) + \beta h_{kl}(1), & d=0 \end{cases} \tag{2.17}$$

Further assume that an estimate of the cover-image histogram $\widehat{h}_{kl}(d)$ is known.

The relative number of changes β can be obtained by minimizing the square error between the observed stego-image histogram $H_{kl}(d)$ and the expected histogram calculated from $\hat{h}_{kl}(d)$:

$$\beta_{kl} = \arg\min_{\beta}[H_{kl}(0) - \hat{H}_{kl}(0)]^2 + [H_{kl}(0) - H_{kl}(0)]^2$$

$$= \arg\min_{\beta}[H_{kl}(0) - \hat{h}_{kl}(0) - \beta h_{kl}(1)]^2 + [H_{kl}(1) - (1-\beta)\hat{h}_{kl}(1) - \beta h_{kl}(2)]^2 \qquad (2.18)$$

Notice that only the histogram values $H_{kl}(0)$ and $H_{kl}(1)$ are involved in calculation of β. The reason is that these two histogram values experience the largest change during embedding due to the shrinkage. Let

$$X = [H_{kl}(0) - \hat{h}_{kl}(0) - \beta\hat{h}_{kl}(1)]^2 + [H_{kl}(1) - (1-\beta)\hat{h}_{kl}(1) - \beta\hat{h}_{kl}(2)]^2 \qquad (2.19)$$

The least square estimation leads to the formula

$$\beta_{kl} = \frac{\hat{h}_{kl}(1)[H_{kl}(0) - h_{kl}(0)] + [H_{kl}(1) - \hat{h}_{kl}(1)][h_{kl}(2) - \hat{h}_{kl}(1)]}{\hat{h}_{kl}(1) + [h_{kl}(2) - \hat{h}_{kl}(1)]^2} \qquad (2.20)$$

The final value of β is

$$\beta = \frac{\beta_{12} + \beta_{21} + \beta_{22}}{3} \qquad (2.21)$$

2.6.4 Estimate of Cover-Image Histogram

To make a precise estimate of the relative number of modifications β, the accurate estimation of the cover-image histogram is absolutely crucial. To obtain the estimate of the cover-image histogram h, the detection first decompress the JPEG stego image, then crop the image in the spatial domain by four columns, and recompress the cropped image using the same quantization table as that of the stego-image. The quantized DCT coefficients can be used to estimate $h_{kl}(d)$. To prevent some spurious non-zero DCT coefficients caused by "discontinuities" at the block boundaries in the cropped image and improve the accuracy of the estimation, a uniform blurring operation is performed before recompressing. The low-pass filter with a 3 x 3 kernel helps to remove possible JPEG blocking artifacts from the cropped image. The kernel B has the form:

$$B=\begin{bmatrix} 0 & e & 0 \\ e & 1-4e & e \\ 0 & e & 0 \end{bmatrix}$$

where e is a user-selected constant.

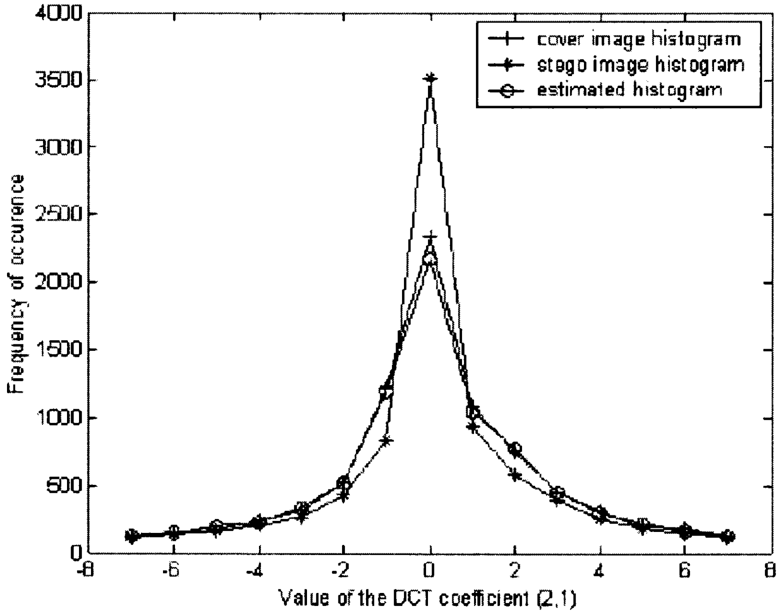


Figure 2.10 The effect of F5 on the histogram of DCT coefficient (2,1).

Figure 2.10 shows that the estimate of the cover-image histogram from the stego-image is very close to the original histogram. It is found that the largest change in histogram values occur in the first two values (d = 0 and d = 1).

Once the relative number of changes β has been determined, the stego image can be distinguished from the cover image. If β is greater than zero, we can claim the image as stego, otherwise the image is considered as the original image. In practice, a zero-value β is seldom obtained even for the cover image due to systematic error. A detection threshold which is slightly greater than zero is set to overcome the initial bias problem. A proper detection threshold will lead to a very small false detection rate and thus increase the positive detection probability.

CHAPTER 3

GENERAL STEGANALYSIS METHODS

3.1 Introduction

3.1.1 General Concept

In Chapter 2 some specific detection methods were presented. The specific detection methods are very reliable and accurate for detecting the targeted embedding algorithms. Since the specific detection methods deal with the targeted steganographic systems, they are only effective for the specific or existing embedding algorithms. Alternatively, the general steganalysis detection methods provide detection regardless of what the steganograpic techniques are used. The general detection methods are more flexible and efficient than the specific detection because they can adjust to new or modified steganographic techniques after the system are properly trained.

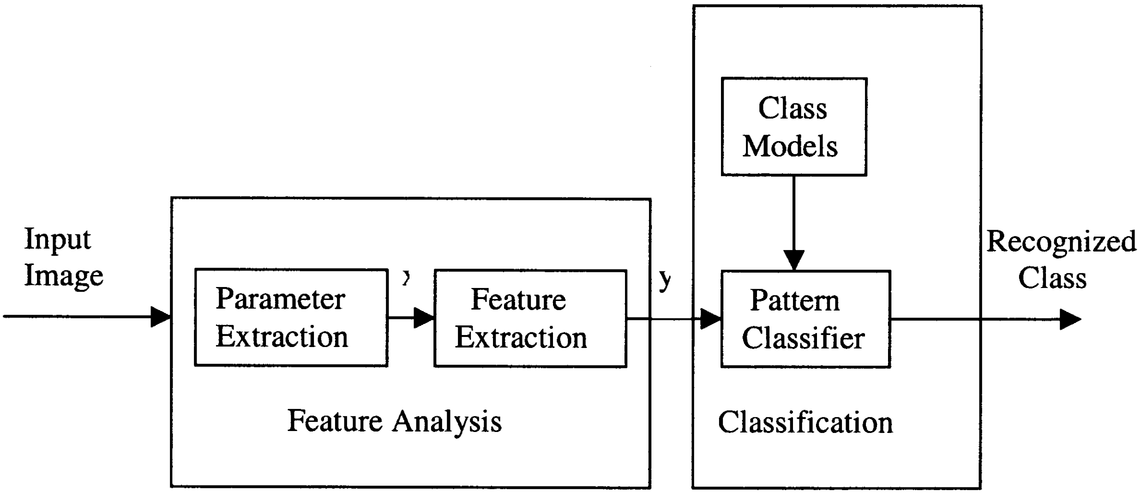


Figure 3.1 The pattern recognition system.

In general, the general detection methods involve the extraction of image features and the classification of the input image into containing the embedded message or not having the hidden message. The general detection method can be treated as a two-class pattern recognition system whose goal is to classify the input image into a given class.

The structure of a conventional pattern recognition system is shown Figure 3.1. The feature analysis includes two steps: parameter extraction and feature extraction. In the parameter extraction step, information relevant with the pattern classification is extracted from the input image to construct a p-dimensional parameter vector x which is transformed to an m-dimensional feature vector $\mathbf{y}$, $m \leq p$. The feature extractor plays an important role in the pattern recognition tasks. For example if the dimensionality of parameter vector is very high, feature reduction is needed for the sake of high computational efficiency and low system complexity.

Once the image features are extracted, classifier can then be used to differentiate between the cover and stego image. There are two parts in the pattern recognition system, namely, training stage and testing stage. Generally, suppose that we are given l exemplars of cover and stego images. Each exemplar consists of a pair: a feature vector $\mathbf{y}_i = R^n$, $i = 1, \ldots, l$, and the associated label $\omega_i \in \{-1, 1\}$. The label $\omega_i = 1$ if $\mathbf{y}_i$ is original, and $\omega_i = -1$ otherwise. In the training stage, the classifier's task is to learn the mapping $\mathbf{y}_i \rightarrow \omega_i$. The training process produces a decision function $f(\mathbf{y}, \alpha)$. The parameter α is adjustable and a particular choice of α generates a "trained machine". The decision function can be applied to determine whether an image is stego or not in the testing stage.

3.1.2 Image Quality Metrics

The first general detection method uses a number of image quality metrics (IQM) to discriminate between cover and stego images. Initially the pixel values are extracted from the original image and passed through a Gaussian lowpass filter. The authors [11] note that embedding of any form results in a message signal being added to the cover signal. Hence, when a blurred image is subtracted from its original, the remaining consists of the high frequency content of the cover summed with the maximum likelihood estimate of the embedded message. If an unmarked image is filtered and subtracted from the original, what is left is only the high frequency content of the image. The idea of this classification is to train a system to distinguish between these two sets of signals (i.e. high frequency vs. high frequency plus embedded message estimate).

The feature vector consists of a number of distances between the filtered image and the original image. These distances include measures such as mean squared error, cross correlation, spectral measures and measures based on the human visual system.

The system is trained using a regression technique. The training step generates a vector of regression coefficients $\boldsymbol{\beta}$ as follows:

$$\boldsymbol{\beta} = (\mathbf{Y}^T\mathbf{Y})^{-1}(\mathbf{Y}^T\omega) \tag{3.1}$$

where $\mathbf{Y}$ represents the matrix of training vectors and ω is the vector of correct labels. In the testing stage, the same metrics are extracted and the resulting feature vector is used in the equation:

$$\hat{\omega} = \mathbf{Y}\boldsymbol{\beta} \tag{3.2}$$

If $\hat{\omega} \geq 0$, the image contains no hidden message, otherwise it is a stego image.

3.1.3 Wavelet Statistics

Another more powerful general detection technique is based on wavelet statistics [8]. The extraction of a feature vector is done by performing an n-level discrete wavelet transform on the image. The classifiers available are Fisher Linear Discriminant (FLD) Analysis [24], Support Vector Machines (SVM) [6], and Neural Network (NN) [27].

SVM is a kernel-based integrated pattern classification algorithm which is very efficient in higher dimensional feature spaces. The classes which are not linearly separable in the original feature space can be linearly separable after being mapped into the higher feature space. The advantage of SVM is that it can handle the classes with complex non-linear decision boundaries.

Neural networks are adaptive structure, consisting groups of neurons operating in parallel. Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific output. Figure 3.2 shows the situation.

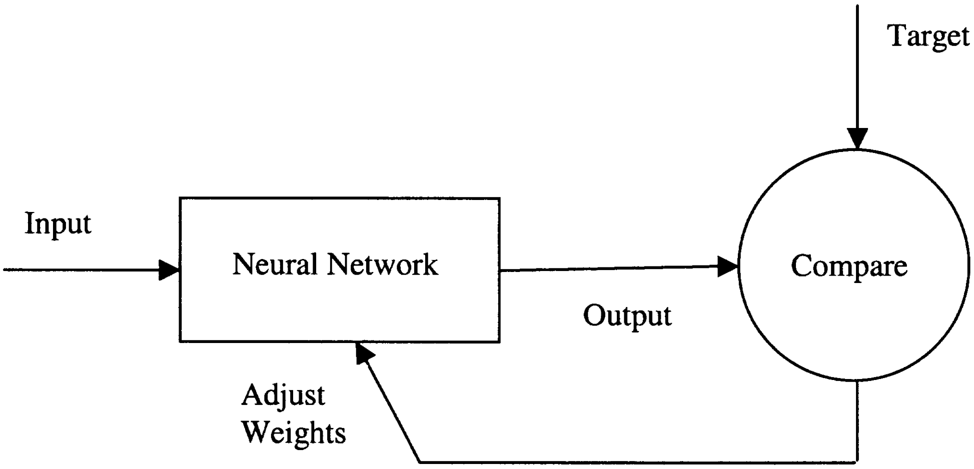


Figure 3.2 The neural network classifier.

In the following, the general detection based on the wavelet statistics is presented.

3.2 Image Features

In the first part of the general detection method, the feature vector is extracted from each investigated image. Although the embedding of messages into an image is often imperceptible to human eye, the statistical regularities of the image are often disturbed. To obtain a set of statistics that forms the feature vector, the wavelet-based decomposition is applied to the image to build a higher–order statistics model.

Wavelet, a waveform which has a limited duration and varying frequency, plays an important role in wavelet transform as the basis function. Wavelets are the foundation of a powerful approach to signal processing and analysis called multiresolution theory which is concerned with the representation and analysis of signal multiresolution.

3.2.1 Discrete Wavelet Transforms

The discrete wavelet transform (DWT) pair for a sequence $f(n)$ is defined by:

$$w_{\varphi}(j_0,k)=\frac{1}{\sqrt{M}}\sum_{n=0}^{M-1} f(n)\varphi(j_0,k,n) \tag{3.3}$$

$$w_{\psi}(j,\, k)=\frac{1}{\sqrt{M}}\sum_{n=0}^{M-1} f(n)\psi(j,k,n) \tag{3.4}$$

for all j_0, j, $k \in \mathbf{Z}$ (the set of integer) and $j \geq j_0$. $\psi(j,k,n)$ is the wavelet function and $\varphi(j_0,k,n)$ is the scaling function. Here j_0 is an arbitrary starting scale, j determines the shape or width of wavelet and scaling function, k determines the position of these functions along n-axis. Normally let $j_0 = 0$ and select M to be a power of 2. The $w_{\varphi}(j_0,k)$ and $w_{\psi}(j,\, k)$ are approximation and detail coefficients, respectively. The approximations are the high-scale, low-frequency components of the signal, and the details are the low-scale, high-frequency components of the signal.

In practice DWT is implemented by applying fast wavelet transform (FWT) which exploits the relationship between the coefficients of the DWT at adjacent scales. The FWT is a computationally effective way to compute the DWT. It resembles the two-band subband coding scheme. Figure 3.3 depicts the structure of the FWT analysis bank.

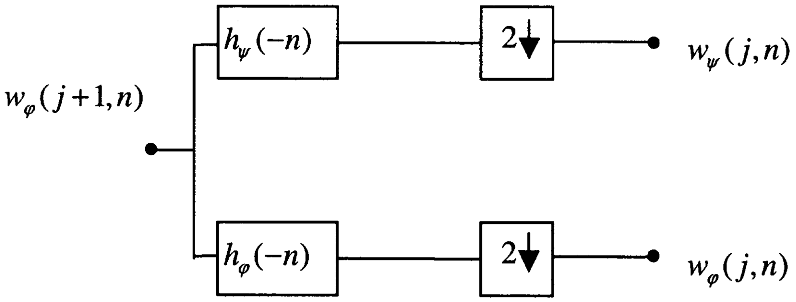


Figure 3.3 FWT analysis bank.

The approximation coefficients $w_\varphi(j+1,n)$ at scale j+1 can be decomposed into the scale j approximation coefficients $w_\varphi(j,n)$ and detail coefficients $w_\psi(j, n)$ via the time-reversed scaling vectors $h_\varphi(-n)$ and wavelet vectors $h_\psi(-n)$, followed by the down-sampling. The scaling function has an association with $h_\varphi(-n)$ which is a low-pass quadrature mirror filter, and the wavelet function is associated with $h_\psi(-n)$ which is a high-pass quadrature mirror filter. The process can be iterated with successive approximations being split in turn, so that one signal is broken down into many lower resolution components. For example, the approximation $w_\varphi(j,n)$ at the scale j can be further split into the approximation coefficients $w_\varphi(j-1,n)$ and the detail coefficients $w_\psi(j-1, n)$ at the scale j-1.

The one-dimensional wavelet decomposition principle can be extended to two-dimensional case, such as images where a two-dimensional scaling function $\varphi(m,n)$, and three two-dimensional wavelets $\psi^H(m,n)$, $\psi^V(m,n)$ and $\psi^D(m,n)$ are required. The scaling function is used to construct the approximations or low-frequency components of the images. The wavelets measure the details or high-frequency components along different directions: $\psi^H(m,n)$ measures the details along the rows, $\psi^V(m,n)$ measures the details along the columns and $\psi^D(m,n)$ responds to the details along diagonals.

The one-dimensional DWT is easily extended to two-dimensional case if the given two-dimensional scaling and wavelet functions are separable, which have the following relationship:

$$\varphi(m,n) = \varphi(m)\varphi(n) \tag{3.5}$$

$$\psi^H(m,n) = \psi(m)\varphi(n) \tag{3.6}$$

$$\psi^V(m,n) = \varphi(m)\psi(n) \tag{3.7}$$

$$\psi^D(m,n) = \psi(m)\psi(n) \tag{3.8}$$

Figure 3.4 shows the process of wavelet-decomposition of the image in one single scale. The two-dimensional DWT is implemented simply by first taking the one-dimensional FWT along each row of the input image $f(m,n)$, and then taking the one-dimensional FWT along each column of these intermediate results. As a result, the $j+1$ scale approximation $w_{j+1}(m,n)$ are broken down into the j scale approximation $w_j(m,n)$ and three sets of detail coefficients: the horizontal details $h_j(m,n)$, vertical details $v_j(m,n)$, and diagonal details $d_j(m,n)$. The decomposition process can be repeated with the successive approximations until one image pixel is left.

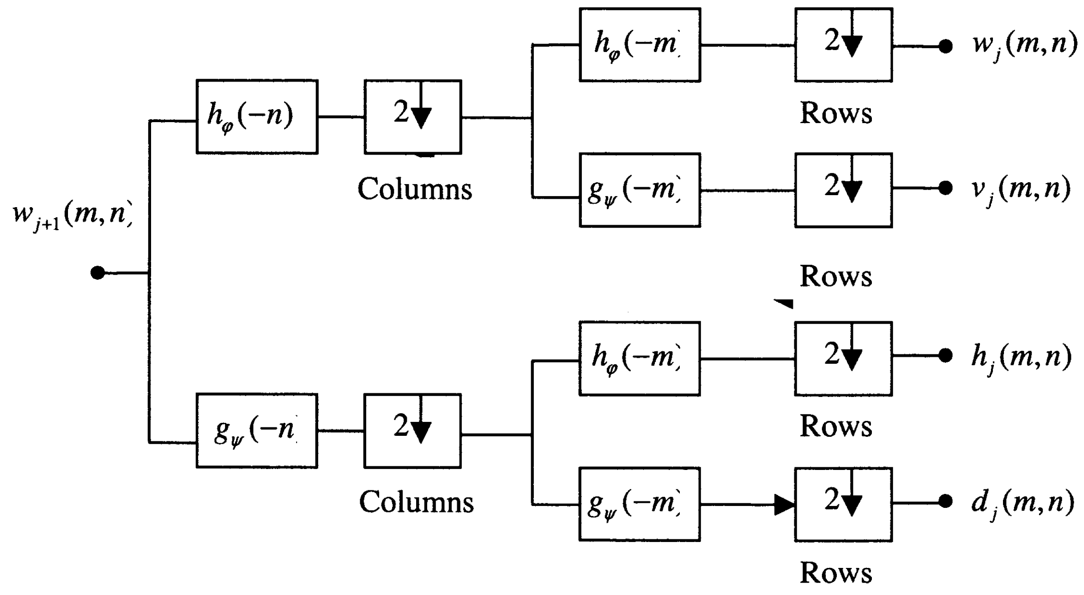


Figure 3.4 The two-dimensional FWT.

In this two-dimensional case, suppose that the image $f(m,n)$ is considered as the $w_J(m,n)$ input with J being an arbitrary starting scale, iterative decomposition of the successive approximations will produce a P scale transform in which scale $j = J - 1, J - 2, \ldots, J - P$. After convolving the rows of the input approximation with the low-pass filter h_φ and high-pass filter g_ψ and then down-sampling the columns, we get two subimages whose horizontal resolutions are one-half of that in the input approximation. Both subimages are then convolved with the same low-pass and high-pass filters and down-sampled along the rows to generate four subimages whose resolutions are one-quarter of the input image. For example, filtering the intermediate subimages containing low-frequency, approximation components with high-pass filter g_ψ and reducing its vertical resolutions by a factor of 2, the vertical detail components $v_j(m,n)$ will be created.

3.2.2 Feature Extraction

After a multi-level discrete two-dimensional wavelet decomposition of an investigated image, the image is decomposed and broken down into four subimages – approximation, vertical, horizontal, and diagonal details. The vertical, horizontal and diagonal high-frequency coefficients at scale $j = 1, 2, \ldots, n$ are denoted as $v_j(m,n)$, $h_j(m,n)$ and $d_j(m,n)$, respectively.

The first set of statistics in the feature vector is the estimates for the first four moments, that is, the mean, variance, skewness and kurtosis of the vertical, horizontal and diagonal coefficients at scales j = 1, 2, . . . , *n-1*. These statistics of a random variable x are defined by

$$\mu_x = E\{x\} \tag{3.9}$$

$$\sigma_x^2 = E\{(x-\mu_x)^2\} \tag{3.10}$$

$$\varsigma_x = E\{(\frac{x-\mu_x}{\sigma_x})^3\} \tag{3.11}$$

$$\kappa_x = E\{(\frac{x-\mu_x}{\sigma_x})^4\} \tag{3.12}$$

where $E\{\bullet\}$ denotes the expectation operator.

The remaining statistics used in the feature vector can be derived from the statistics of the prediction error of the detail coefficients. A specific detail coefficient is correlated to its four neighboring coefficients, the corresponding coefficients in the coarser scale at the same orientation and the details coefficients of other orientations and scales. The linear predictor for the magnitude of the vertical detail coefficients $v_j(m,n)$ is given by:

$$\tilde{v}_j(m,n) = w_1 v_j(m-1,\ n) + w_2 v_j(m+1,n) + w_3 v_j(m,n-1) + w_4 v_j(m,n+1)$$

$$+ w_5 v_{j+1}(m/2,n/2) + w_6 d_j(m,n) + w_7 d_{j+1}(m/2,n/2) \tag{3.13}$$

where w_k is the predictor weight. So a vertical detail coefficient at scale j can be estimated by the spatial neighbors $v_j(m-1,n)$, $v_j(m+1,n)$, $v_j(m,n-1)$ and $v_j(m,n+1)$, the neighbors at scale j+1 $v_{j+1}(m/2,\ n/2)$ and $d_{j+1}(m/2,\ n/2)$, and the orientation neighbor $d_j(m,n)$ at scale j. Equation (3.13) can be written in the matrix form:

$$\tilde{\mathbf{V}} = Q\mathbf{w} \tag{3.14}$$

where $\tilde{\mathbf{V}}$ is the column vector containing the estimates of coefficient magnitudes of $v_j(m,n)$, the column vector w = $\begin{bmatrix} w_1 & \cdots & w_7 \end{bmatrix}^T$, and the rows of the matrix Q contain the neighboring coefficients.

Suppose that $\mathbf{V}$ is the column vector consisting of coefficient magnitudes $v_j(m,n)$. By minimizing the mean squared error of the linear predictor of coefficient magnitude, the optimal prediction weights is

$$w_{opt} = (Q^T Q)^{-1} Q^T V \tag{3.15}$$

The log error of the linear predictor is defined by

$$e_{v_j}^{\log}(m,n) = \log_2(|v_j(m,n)|) - \log_2(|\tilde{v}_j(m,n)|) \tag{3.16}$$

where $\tilde{v}_j(m,n)$ is the optimal estimate of $v_j(m,n)$. The mean, variance, skewness, and kurtosis of the log error of the detail coefficients form another set of the feature vector. The horizontal detail h_j and diagonal detail d_j can be predicted in the similar way. The linear predictor for h_j has the following form:

$$\tilde{h}_j = w_1 h_j(m-1,n) + w_2 h_j(m+1,n) + w_3 h_j(m,n-1) + w_4 h_j(m,n+1)$$

$$+ w_5 h_{j+1}(m/2,n/2) + w_6 d_j(m,n) + w_7 d_{j+1}(m/2,n/2) \tag{3.17}$$

Similarly the linear predictor for d_j is

$$\tilde{d}_j = w_1 d_j(m-1,n) + w_2 d_j(m+1,n) + w_3 d_j(m,n-1) + w_4 d_j(m,n+1)$$

$$+ w_5 d_{j+1}(m/2,n/2) + w_6 h_j(m,n) + w_7 v_j(m,n) \tag{3.18}$$

The same error statistics for h_j and d_j are computed. There are totally 12 ? $(l - 1)$ error statistics, l is the decomposition level. The error statistics are combined with the coefficient statistics to form a 24 ? $(l - 1)$-dimensional feature vector. Given the feature vector, the classification can be performed to determine whether an image contains an embedded message.

3.3 Fisher Linear Discriminant Analysis

3.3.1 FLDA Classification

The Fishier linear discriminant analysis is a non-parametric classification. It reduces the dimension of the feature space by projecting a feature vector to a lower dimensional space. Suppose that there are n training samples $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ in a d-dimensional feature space, n_1 in subset χ_1 corresponds to class ω_1 and n_2 in subset χ_2 corresponds to class ω_2. The goal is to find a vector $\mathbf{w}$, and project the n samples on this axis denoted by y:

$$y = \mathbf{w}^T \mathbf{x} \tag{3.19}$$

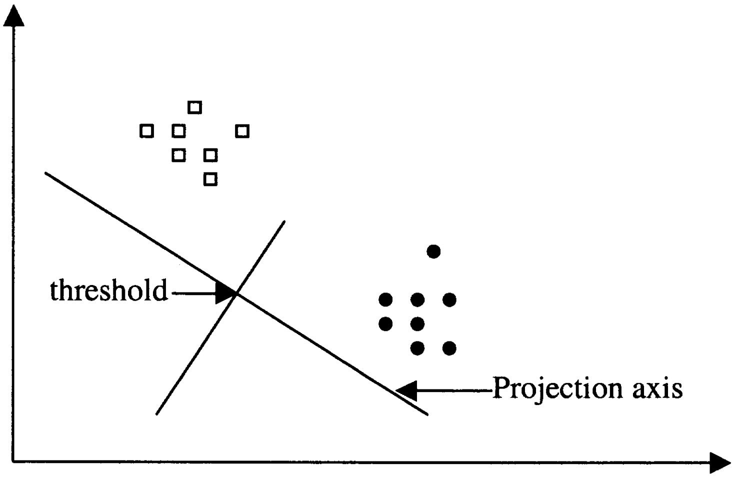


Figure 3.5 Two-class FLD.

so that the projected samples are well separated. Now we give the definition used in the two-class FLD.

The sample mean for class ω_i:

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \chi_i} \mathbf{x}, \quad i = 1, 2 \tag{3.20}$$

The scatter matrix for class ω_i:

$$S_i = \sum_{\mathbf{x} \in \chi_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T, \ i = 1, 2 \tag{3.21}$$

The within-class scatter matrix:

$$S_W = S_1 + S_2 \tag{3.22}$$

The between–class scatter matrix:

$$S_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \tag{3.23}$$

The vector $\mathbf{w}$ that minimizes the between–class distance and maximizes the within-class scatter is the maximal generalized eigenvalue-eigenvector of S_B and S_W, i.e.

$$S_B \mathbf{w} = \lambda S_W \mathbf{w} \tag{3.24}$$

Once the FLD projection axis is determined from the training set, a novel exemplar $\mathbf{z}$ from the testing set is classified by first projecting into the same subspace, $\mathbf{z}^T\mathbf{w}$. Then the class to which this exemplar belongs is determined via a simple threshold (See Figure 3.5).

3.3.2 Performance

A two–class is employed in [8] to classify images into containing or not containing a hidden image. Each image is characterized by its feature vector as described above. The two-class FLD is trained separately to classify the LSB embedding in the JPEG, GIF, and TIFF images. The classifier provides accurate detection when the message size is significant.

3.4 Support Vector Machine Classification

3.4.1 Linear Separable SVM

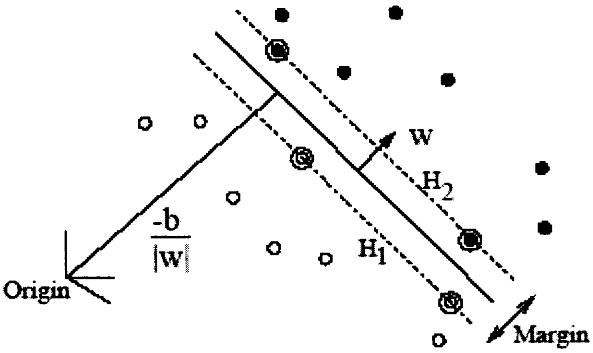


Figure 3.6 Linear separating hyperplane for the separable case.

The support vector machine (SVM) is more flexible than FLD analysis. The SVM can handle not only linear case but also non-linear case. The simplest and most straightforward SVM is the linear machine trained on separable data. Denote the training data pair $\{\mathbf{y}_i, \omega_i\}, i=1,\ldots,l$, $\mathbf{y}_i$ is the feature vector consisting of the image statistics, and $\omega_i = 1$ for images not containing a hidden message, and $\omega_i = -1$ for images with a hidden message. For the linearly separable case, the SVM classifier simply searches for a hyperplane that separates the positive pattern from the negative pattern. Figure 3.6 shows a two-class SVM. The points lying on the hyperplane satisfy

$$\mathbf{w}^t \mathbf{y}_i + b = 0 \tag{3.25}$$

where **w** is normal to the hyperplane, $|b|/||\mathbf{w}||$ is the perpendicular distance from the origin to the hyperplane, and $||\mathbf{w}||$ is the Euclidean norm of **w**. Suppose that d_+ and d_- are the distances from the hyperplane to the closest positive and negative exemplars, respectively. Define the "margin" of a separating hyperplane to be $d_+ + d_-$.

The linear support vector algorithms can be formulated as follows: if a separating hyperplane exists, then all of the training data satisfy the following constraints:

$$\mathbf{w}^t \mathbf{y}_i + b \geq 1 \quad \text{if } \omega_i = +1 \tag{3.26}$$

$$\mathbf{w}^t \mathbf{y}_i + b \leq -1 \quad \text{if } \omega_i = -1 \tag{3.27}$$

They can be combined into the following inequalities:

$$\mathbf{w}^t \mathbf{y}_i + b \ -1 \geq 0 \quad \forall i \tag{3.28}$$

The points which satisfy the equality in (3.26) lie on the hyperplane H_1: $\mathbf{w}^t \mathbf{y}_i + b = 1$. The perpendicular distance from the origin to the hyperplane H_1 is $|1\text{-}b|/||\mathbf{w}||$. Similarly, the points on the hyperplane H_2 satisfy the equality: $\mathbf{w}^t \mathbf{y}_i + b = -1$. The perpendicular distance from the hyperplane H_2 to the origin is $|\text{-}1\text{-}b|/||\mathbf{w}||$. So both d_+ and d_- are equal to $1/||\mathbf{w}||$. The margin ($=d_+ + d_-$) is $2/||\mathbf{w}||$.

Therefore, by minimizing $||\mathbf{w}||^2$ subject to the constraints in Equation (3.28) we can find the pair of hyperplanes which give the maximum margin. The optimization problem is reformulated using Lagrange multipliers for the ease of computation. Recall that the Lagrangian is formed by subtracting the constraint equations multiplied by positive Lagrange multipliers from the objective function. This yields the following Lagrangian:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \omega_i (\mathbf{w}^t \mathbf{y}_i + b) + \sum_{i=1}^{l} \alpha_i \tag{3.29}$$

where α_i is the Lagrange multiplier and $\alpha_i \geq 0$.

This cost function of L_P must be minimized with respect to **w**, *b*, while requiring that the derivatives of L_P with respect to all α_i are zeros. Because this is a convex quadratic programming problem, we can solve its dual problem to yield the same results of **w**, *b*, and α_i. In the dual problem, the cost function L_D is maximized with respect to α_i while requiring that the derivatives of L_D with respect to **w** and *b* are zeros. Note that the subscript P and D in the cost function: P for primal and D for dual. Setting the derivatives of L_D with respect to **w** equal to zero yields:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i \mathbf{y}_i \omega_i \tag{3.30}$$

and setting the derivatives of L_D with respect to *b* equal to zero gives

$$\sum_{i=1}^{l} \alpha_i \omega_i = 0 \tag{3.31}$$

Substituting these two constraints into Equation (3.29) to give

$$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j \mathbf{y}_i^t \mathbf{y}_j \omega_i \omega_j \tag{3.32}$$

The training amounts to maximizing L$_D$ with respect to α_i. If the classifier is linear separable, the solution exists. That means for every training point, there is a Lagrange multiplier α_i. Only those points corresponding to α_i > 0 make contribution to the training and lie on one of the hyperplanes H$_1$ and H$_2$. We call them “support vectors”. In other words, even if all other training points having α_i = 0 are discarded, and the training is repeated, the same separating hyperplane will be obtained. Once the α_i is determined, the normal to the hyperplane **w** can be calculated based on the equation (3.30). The parameter *b* is computed by using the Karush-Kuhn-Tucker condition as follows:

$$b = \frac{1}{l}\sum_{i=1}^{l} (\omega_i - \mathbf{w}^t \mathbf{y}_i) \tag{3.33}$$

Once the SVM has been trained, the novel exemplar **z** from the testing set of images can be classified from the **w** and *b*. If $\mathbf{w}^t \mathbf{z} + b$ is greater than or equal to zero, the image is classified as not containing a hidden message, otherwise it is classified as having a hidden message.

3.4.2 Linear Non-Separable SVM

In the non-separable case, the training data crosses the hyperplane H$_1$ or/and H$_2$, as illustrated in the Figure 3.7. To handle such situation, the initial constraints expressed in Equation (3.26) and (3.27) should be relaxed. By introducing positive slack variables $\xi_i, i = 1, 2, \ldots, l$ the modified constraints will be given as follows:

$$\mathbf{w}^t\mathbf{y}_i + b \geq 1 - \xi_i \quad \text{if } \omega_i = +1 \tag{3.34}$$

$$\mathbf{w}^t\mathbf{y}_i + b \leq -1 + \xi_i \quad \text{if } \omega_i = -1 \tag{3.35}$$

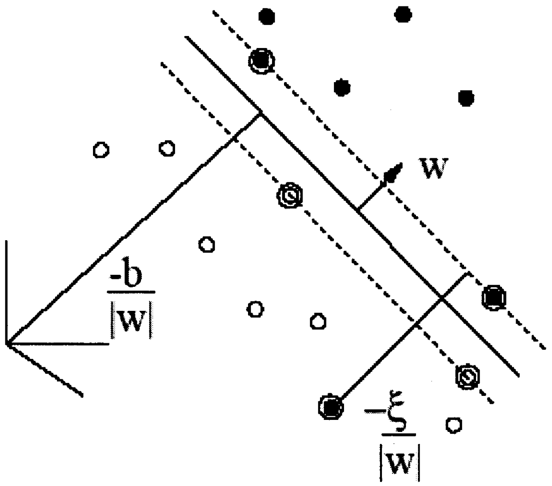


Figure 3.7 Linear separating hyperplane for non-separable data.

A training point lying on the wrong side of the separating hyperplane will have a corresponding ξ_i whose value exceeds unity. The linear non-separable training amounts to minimizing the extra cost $\sum_i \xi_i$, and simultaneously maximizing the margin. The objective function will be modified to include the extra cost and it is equal to $\frac{1}{2}\|\mathbf{w}\|^2 + C(\sum_i \xi_i)^k$, where C is a user-selected parameter controlling the relative penalty for the training error, and *k* is a positive integer. Minimizing the cost function is still a quadratic programming problem which can be expressed in term of its dual problem. When *k* is chosen to 1, neither the slack variables nor their Lagrange multipliers will appear in the dual problem with the same expression as Equation (3.32). Rewrite the equation as follows:

$$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j \mathbf{y}_i^t \mathbf{y}_j \omega_i \omega_j \tag{3.36}$$

By maximizing the cost function subject to

$$0 \le \alpha_i \le C \tag{3.37}$$

$$\sum_i \alpha_i \omega_i = 0 \tag{3.38}$$

The normal to the separating hyperplane will be given by

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \mathbf{y}_i \omega_i \tag{3.39}$$

where *N* is the number of support vectors. Just as before, the parameter *b* is determined using the KKT conditions.

3.4.3 Non-Linear SVM

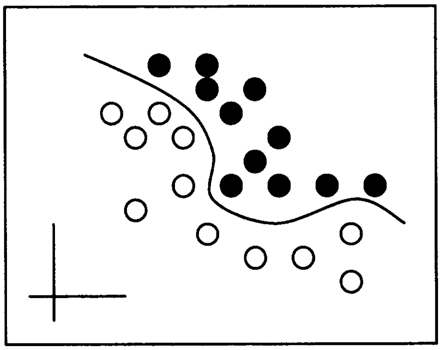


Figure 3.8 Non-linear SVM.

In the case shown in Figure 3.8, a non-linear hyperplane will provide more accurate classification. To realize the non-linear SVM, the training data $\mathbf{y}_i$ in the space *L* is first mapped into the higher (possibly infinite) dimensional Euclidean space *H*, using the mapping $\Phi : L \rightarrow H$. The linear SVM is then employed with the mapped training data

$\Phi(\mathbf{y}_i)$ in the space *H*. Following the same procedures as described in the previous section yields the cost function:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j \Phi^t(\mathbf{y}_i) \Phi(\mathbf{y}_j) \omega_i \omega_j \tag{3.40}$$

Now the cost function to be maximized with respect to α_i depends only on the inner products $\Phi(\mathbf{y}_i)$ and $\Phi(\mathbf{y}_j)$ in the space *H*. It is not easy to explicitly compute $\Phi(\mathbf{y}_i)$. To avoid this burden, a kernel function $K(\mathbf{y}_i \mathbf{y}_j)$ is introduced such that

$$K(\mathbf{y}_i \mathbf{y}_j) = \Phi^t(\mathbf{y}_i) \Phi(\mathbf{y}_j) \tag{3.41}$$

By replacing $\Phi^t(\mathbf{y}_i), \Phi(\mathbf{y}_j)$ with $K(\mathbf{y}_i \mathbf{y}_j)$ everywhere in the training algorithm, a linear SVM will be produced in the higher dimensional space *H*.

Once the hyperplane parameters **w** and *b* are determined, we can classify the new exemplar by deciding to which side of the separating hyperplace it belongs. Because **w** lives in the space *H*, the new exemplar needs to be mapped into *H* before performing classification. Denote the new exemplar as **z**, we compute the quantity $f(\mathbf{z}) = \mathbf{w}^t \Phi(\mathbf{z}) + b$. The kernel function may be employed to avoid the computation of Φ.

$$\begin{aligned} f(\mathbf{z}) &= \mathbf{w}^t \Phi(\mathbf{z}) + b \\ &= \sum_{i=1}^{l} \alpha_i \omega_i \Phi^t(\mathbf{y}_i) \Phi(\mathbf{z}) + b \\ &= \sum_{i=1}^{l} \alpha_i \omega_i K(\mathbf{y}_i, \mathbf{z}) + b \end{aligned} \tag{3.42}$$

If $f(\mathbf{z})$ is greater than or equal to zero, the exemplar is classified as not containing a hidden message, otherwise it is classified as having a hidden message.

3.4.4 The Performance

The SVM classifier has a remarkable performance for detecting whether an image has a message embedded in the LSB plane. Farid *et al.* [9] showed that linear non-separable SVMs have similar classification accuracy to the FLD, and non-linear SVM performs even better.

3.5 Neural Networks Classification

3.5.1 Feed-Forward Neural Networks

The FLD and SVM are very effective in dealing with the LSB embedding. Still based on the features described as above, the feed-forward neural network with back propagation learning is proposed to classify images.

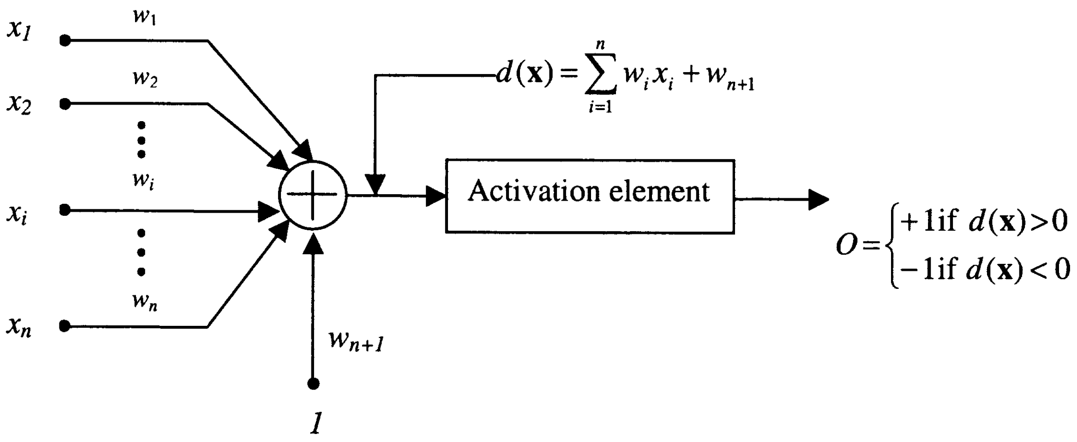

x_1
w_1
x_2
w_2
x_i
w_i
x_n
w_n
w_{n+1}
1
$d(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + w_{n+1}$
Activation element
$O = \begin{cases} +1 \text{if } d(\mathbf{x}) > 0 \\ -1 \text{if } d(\mathbf{x}) < 0 \end{cases}$

Figure 3.9 Perceptron.

Neural network is an adaptive structure which consists of single elements operating in parallel. Such element, called perceptron, is a basic device of the neural network. Figure 3.9 shows the perceptron model for two pattern classes. The response of the perceptron is based on a weighted sum of its inputs, that is,

$$d(\mathbf{x})=\sum_{i=1}^{n} w_i x_i + w_{n+1} \tag{3.43}$$

It is a linear decision function with respect to the components of the feature vectors. The coefficients w_i, $i = 1, 2, \ldots, n+1$ called weights (or bias for w_{i+1}) modify the inputs before they are summed and fed into the threshold element. The activation element that maps the weighted sum of its inputs into the final response of the device is called activation function.

When $d(\mathbf{x}) > 0$, the activation function gives the output O equal to +1, indicating that the pattern **x** was recognized as belonging to class ω_1. The reverse is true when $d(\mathbf{x}) < 0$. When $d(\mathbf{x}) = 0$, **x** lies on the decision surface that separates the two pattern classes, giving an indeterminate condition. The decision surface can be represented by the equation of a hyperplane in n-dimensional pattern space, which is obtained by setting Equation (3.43) equal to zero:

$$d(\mathbf{x})=\sum_{i=1}^{n} w_i x_i + w_{n+1} = 0 \tag{3.44}$$

or

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{n+1} = 0 \tag{3.45}$$

Geometrically, the first n coefficients establish the orientation of the hyperplane, whereas the last coefficient, w_{n+1}, is proportional to the perpendicular distance from the origin to the hyperplane. If $w_{n+1}=0$, the hyperplane goes through the origin of the pattern space. Similarly, if $w_j = 0$, the hyperplane is parallel to the x_j axis.

The output of the activation element in Figure 3.9 depends on the sign of $d(\mathbf{x})$. We could test the summation part of Equation (3.43) against the term w_{n+1}. In such case, the output of the system would be

$$O=\begin{cases}+1, \text{ if } \sum_{i=1}^{n} w_i x_i > -w_{n+1} \\ -1, \text{ if } \sum_{\mathrm{i=1}}^{\mathrm{n}} w_i x_i < -w_{n+1}\end{cases} \tag{3.46}$$

Another formation commonly used is to augment the feature vectors by appending an additional $(n + 1)$st element, which is always equal to 1, regardless of class membership. That is, an augmented feature vector **y** is created from a feature vector **x** by letting $y_i = x_i$, $i = 1, 2, \ldots, n$ and appending the additional element $y_{n+1} = 1$. Equation (3.43) then becomes

$$d(\mathbf{x})=\sum_{i=1}^{n+1} w_i y_i = \mathbf{w}^{\mathrm{T}}\mathbf{y} \tag{3.47}$$

where $\mathbf{w} = (w_1, w_2, \ldots, w_n, w_{n+1})$ is the weight vector, and the augmented feature vector is $\mathbf{y} = (y_1, y_2, \ldots, y_n, 1)^T$. Regardless of the formulation used, however, the key problem is to find **w** by using a given training set of pattern vectors from each of two classes.

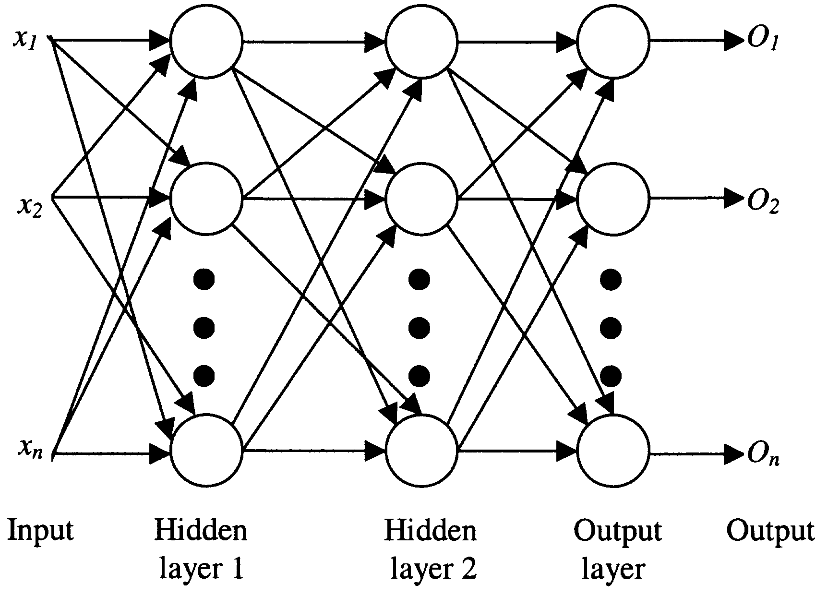


Figure 3.10 Three-layer neural networks.

The elementary perceptron is only able to solve linearly separable classification problem. For the linearly non-separable or non-linearly input patterns that require greater discrimination, multilayer artificial networks are used to find the decision function. The key to the multilayer artificial network is the learning algorithms that allow the training of the weights on the hidden layers whose neurons are not directly connected to the inputs or the outputs. Figure 3.10 shows the architecture of a three-layer neural network. Each node represents a perceptron in which the activation element normally used is the sigmoid. The simplicity of the derivative of the sigmoid provides a nice learning law. It was believed that three layers were the most required for any arbitrary classification problem.

The reason to use multilayer networks is to allow the formation of a complex decision regions in the feature space. The main problem with a multilayer network is to adjust the weights in the hidden layers and output layer. The most commonly used algorithm is called the back propagation learning rule which will be discussed in the following section.

3.5.2 Training by Back Propagation

Back propagation was created by generalizing the delta training rule to multilayer networks. The training algorithm is an iterative gradient algorithm designed to minimize the mean square error between the actual output and the targeted output at the output perceptron.

The delta rule with N nodes on the output layer of a network is described below. The mean square error has to be summed over all N nodes. The idea is to perform a gradient descent on the error which is considered as a function of the weights. All the weights for both hidden and output nodes need to be taken into account. The hidden nodes are in the intermediate layers to which we do not have direct access during the training. The output nodes are the ones which tell us the net's response to an input and to which we may show a target value during training.

For the output nodes, the delta rule is defined in the following equation:

$$\Delta w_i^j = \alpha\sigma'(a^j)(t^j - y^j)x_i^j \tag{3.48}$$

The superscript means that the *j*th node is being described. Note that only the part containing reference to *j*th node can affect the gradient of the error on the *j*th node with respect to a weight. In the right hand side of Equation (4.48), the term $(t^j - y^j)$ is the error between the target response and the actual response on the *j*th node. The term $\sigma'(a^j)$ relates to how quickly the activation can change the output (and hence the error). The large it is, the more change we can expect. The factor x_i^j is related to the effect that the *i*th input has on the activation. When it is zero, the input is nothing to do with the error and thus the weight change should also be zero. On the other hand, when it is large, the *i*th input is responsible for the error and so the weight needs to be changed by a corresponding larger amount.

In summary, x_i^j tells us how much the *i*th input contributes to the activation; $\sigma'(a^j)$ tells us what rate of change the activation has, and $(t^j - y^j)$ is the error on the *j*th node. Therefore it is reasonable that the product of these terms gives us something that is a measure of the slope of the error with respect to the weight w_i^j. Let the product to be denoted as follows:

$$\delta^j = \sigma'(a^j)(t^j - y^j) \tag{3.49}$$

The delta rule for the output nodes may be rewritten:

$$\Delta w_i^j = \alpha \delta^j x_i^j \tag{3.50}$$

Now extend to two-layer network, and consider the *k*th hidden node. The problem with the assignment of a set of weight changes to this type of node is related to how much this node is responsible for the error. For the *i*th input to the *k*th hidden node, the value of the input will play a similar role just as before so we may write:

$$\Delta w_i^k = \alpha \delta^k x_i^k \tag{3.51}$$

To determine the factor δ^k, let us first consider just a single output from the *k*th hidden node to the *j*th output node (See Figure 3.11).

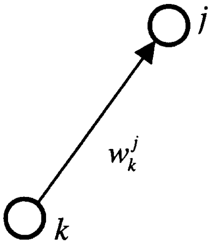


Figure 3.11 *k*th hidden and *j*th output nodes.

How much influence the hidden node has on the error is decided by two things: first, to what extent it can affect the output of *j*th node and, via this, how much the output of *j*th node influence the error. The influence of *k* on *j* is through the weight w_k^j, and the contribution made by the node *j* to the error is expressed in the 'delta' for that node δ^j. It is therefore intuitive to expect the product of w_k^j with δ^j in the expression for δ^k. On the other hand, considering that the node *k* may give output to several output nodes, the contributions to the error from all of these must be taken into account. Thus, these products must be summed over all output nodes having connection to the *k*th hidden node. Finally, the factor $\sigma'(a^k)$ will occur for exactly the same reason that it did for the output nodes. This will give the following expression for the δ^k :

$$\delta^k = \sigma'(a^k)\sum_{j\in I_k} \delta^j w_k^j \tag{3.52}$$

where I^k is the set of nodes with an input from the *k*th hidden node. The weight changes in the hidden nodes are calculated by substituting this into Equation (3.51).

In summary, the process starts with an arbitrary (but not all equal) set of weights throughout the network. Then at any iteration, there are two basic phases involved. In the first phase, a training feature vector is presented to the network and is allowed to propagate through the layers to evaluate the output for each node. The outputs of the nodes in the output layer are then compared against their desire responses, to generate the error. The second phase involves a backward pass through the network. The appropriate error signal is passed to each node and the corresponding weight changes are made. In a successful training, the network error decreases with the number of iterations and the procedure converges to a stable set of weights that exhibit only small fluctuations with additional training.

3.5.2 Simulation Results

The images used in our experiment are taken from Coreldraw. Figure 3.12 shows sample images. Each BMP image is first converted to gray-scale (gray = 0.299R + 0.587G + 0.114B). Statistics are collected from 139 such images. A four-level, three-orientation (vertical, horizontal and diagonal) QMF pyramid is performed for each image, from which a 72-dimensional feature vector of coefficient and error statistics is constructed.

Figure 3.12 Sample images.

The message was embedded into the set of images using quantization index modulation (QIM). The QIM steganographic technique hides information by modulating the rounding values of the JPEG DCT coefficients in the quantization process. When the message bit takes on value of one, it is embedded in the JPEG image by modulating the rounding choice up in the DCT coefficient. When the message bit is zero, it is hidden by modulating the rounding choice down in the DCT coefficient. After the message is embedded into the cover image, the same transformation, decomposition, and collection of statistics as described above is performed.

A three-layer feedforward neural network is employed. There are 60 neurons in the first hidden layer and 45 neurons in the second hidden layer. Only one neuron is needed in the output layer since this is two-class classification. The activation function for the neuron of hidden layer is sigmoid (See Figure 3.13 (right)). The neuron in the output layer adopts linear activation function (See Figure 3.13 (left)).

a
+1
n
0
-1
a
+1
n
0
-1

Figure 3.13 The activation function.

The neural network was implemented using Matlab Neural Network Toolbox. The training set consists of a random subset of images from the 139 original and stego images. The trained neural network is then used to classify all of the remaining previously unseen images. If the output is greater than or equal to zero, the image is classified as not containing hidden message. Otherwise it is classified as having the message embedded. The number of inputs are various in the training. We used 24, 28, 35, 47 and 35 pairs of images to train the neural network, respectively. Shown in Table 3.1 are classification results. It's seen that the classification rate increases when the size of training set is up. In any case, the classification rate is above 80%. This performance is quite satisfactory.

Table 3.1 Classification Accuracy

Training set	Testing set	cover image detection rate	stego image detection rate	overall detection rate
48	230	81.74%	82.61%	82.17%
56	222	86.49%	87.39%	86.94%
70	208	86.54%	90.39%	88.46%
94	184	89.13%	91.30%	90.22%
140	138	92.75%	89.86%	91.31%

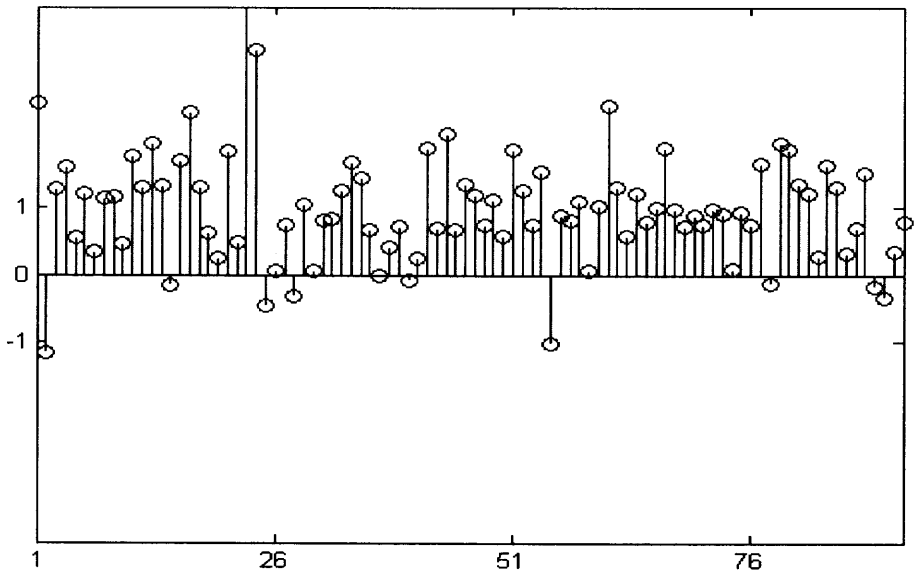


Figure 3.14 The classification results for cover images.

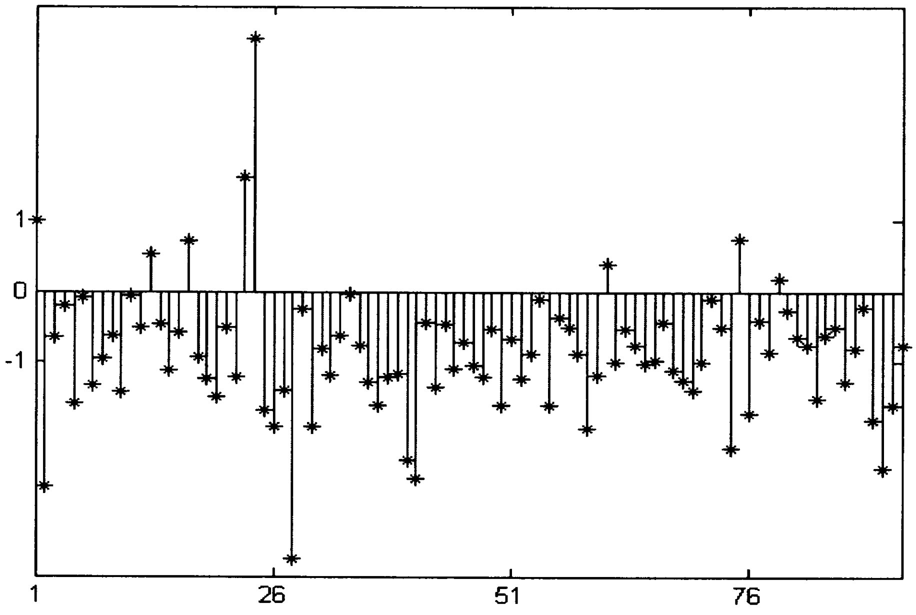


Figure 3.15 The classification results for stego images.

When 94 images were used to train the neural network, the classification results for the remaining 184 images are graphically represented in Figures 3.14 and 3.15. The x axis is the image index and the y axis is the classifier output. The circle represents the classifier output when the input is the original cover image. If the input is stego image, the classifier output is denoted as the asterisk. In the Figure 3.14, there are 10 miss detection (the points below the x axis). In the Figure 3.15, the number of miss detection (above the x axis) is 8.

CHAPTER 4

CONCLUSION

In this paper, the steganalysis detection methods in the literature are surveyed and studied. Generally the detection methods can be classified into two categories: specific and general steganalysis detection. Specific detection methods deal with targeted steganographic systems. A significant body of work in this area focuses on the detection of LSB embedding techniques. General detection methods provide detection of any kind of steganographic techniques after proper training on sufficient number of original and stego images. Specific detection methods will always provide more reliable and accurate results. Nevertheless, general detection methods are more flexible and efficient because they can adjust to new or unknown steganographic systems.

In addition to the detection of the embedded message, the goal of steganalysis includes: (1) estimate the length of the embedded message; (2) estimate the location(s) occupied by the hidden data; (3) estimate the stego key used by the embedding algorithm; and (4) extract the hidden information. In the future, more works are needed to tackle these issues.

The battle between steganography and steganalysis is never-ending. New, more sophisticated steganographic techniques will require more powerful detection methods. The existing steganalysis methods form a very small tip of a very big iceberg that hides exciting and challenging research for the years to come.

REFERENCES

1. A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," in *Lecture Notes in Computer Science*, **1768**, pp. 61-75, Springer-Verlag, Berlin, 2000.

2. A. Westfeld and A. Pfitzmann, "High capacity despite better steganalysis (F5-A Steganographic Algorithm)," in *Lecture Notes in Computer Science*, **2137**, pp. 289-302, Springer-Verlag, Berlin, 2001.

3. Andrew D. Ker, "quantitative evaluation of pairs and RS steganalysis," *Security, Steganography, and Watermarking of Multimedia Contents VI*, E. J. Delp III and P. W. Wong, editors, *Proc. of SPIE*, **5306**, pp. 13-22, 2004.

4. A. Westfeld, *F5*, Software available at wwwrn.inf.tu-dresden.de/westfeld/f5.

5. B. Chen and G. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Trans. On Information Theory*, **47,** pp. 1423-1443, May 2001.

6. C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, **2**, pp. 121-167, 1998.

7. D. Upham, *Jsteg*, Software available at ftp.funet.fi**.**

8. H. Farid, "Detecting hidden messages using higher-order statistical models," *in Proc. Of the International Conference on Image Processing, IEEE, 2002.*

9. H. Farid and S. Lyu, "Detecting hidden messages using higher-order statistics and support vector machines," *pre-proceedings 5th Information Hiding Workshop*, Netherlands, Oct. 7-9, 2002.

10. Help documents in MATLAB Version 6.5

11. I. Avcibas, N. Memon, and B. Sankur, "Steganalysis using image quality metrics," *IEEE Trans. on Image Processing*, **12,** pp. 221-229, February 2003.

12. J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in color and grayscale images," *Proc. ACM Workshop on Multimedia and Security*, pp. 27-30, 2001.

13. J. Fridrich, and M. Goljan, "Practical steganalysis of digital images – state of the art," *Security and Watermarking of Multimedia Contents IV*, E. J. Delp III and P. W. Wong, editors, *Proc. of SPIE*, **4675**, pp. 1-13, 2002.

14. J. Fridrich, M. Goljan, and D. Hogea, "Steganalysis of JPEG Images: breaking the F5 algorithm," *pre-proceedings of 5th Information Hiding Workshop*, Netherlands, Oct. 7-9, 2002.

15. J. Fridrich, M. Goljan, and D. Hogea, "Attacking the OutGuess," *Proc. of ACM: Special Section on Multimedia Security and Watermarking*, Juan-les-Pins, France 2002.

16. J. Fridrich, M. Goljan, and R. Du, "Steganalysis based on JPEG compatibility," *Multimedia Systems and Applications IV*, A. G. Tescher, B. Vasudev, and V. M. Bove, Jr, Editors, *Proc. of SPIE*, **4518**, pp. 275-280, 2002.

17. J. Fridrich, M. Goljan, and D. Soukal, "Higher-order statistical steganalysis of palette images," *Security and Watermarking of Multimedia Contents V*, E. J. Delp III and P. W. Wong, editors, *Proc. of SPIE*, **5020**, pp. 178-190, 2003.

18. Mark T. Hogan, Guenole C. M. Silvestre, and Neil J. Hurley, "Performance evaluation of blind steganalysis classifiers," *Security, Steganography, and Watermarking of Multimedia Contents VI*, E. J. Delp III and P. W. Wong, editors, *Proc. of SPIE*, **5306**, pp. 58-69, 2004.

19. Neil F. Johnson and Sushil Jajodia, "Steganalysis of images created using current steganography software," in *Lecture Notes in Computer Science*, **1525**, pp. 273-289, Springer-Verlag, Berlin, 1998.

20. Neil F. Johnson and Sushil Jajodia, "Steganalysis: the investigation of hidden information," *IEEE Information Technology Conference*, Syracuse, New York, USA, September 1st - 3rd, 1998: 113-116.

21. Neil F. Johnson, Z. Duric and S. Jajodia, *Information Hiding: Steganography and Watermarking – Attacks and Countermeasures*, Kluwer Academic Publishers, Boston Dodrecht London, 2000.

22. Niels Provos, *OutGuess*, http://www.outguess.org/, August 1998.

23. R. Chandramouli and K. Subballakshmi, "Active steganalysis of spread spectrum image steganography," *Proc. Of ISCAS'03*, **3,** pp. 830-833, May 2003.

24. R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, J. Wiley & Sons, first edition, 1973.

25. R. Tzschoppe, R. Bauml, J. Huber, and A. Kemp, "Steganographic system based on higher order statistics," *Security and Watermarking of Multimedia Contents, Proc. Of Electronic Imaging*, **5020,** SPIE, January 2003.

26. R. Machado, *Ezstego*, http://www.fqa.com/romana.

27. Rafael Gonzalez and Richard Woods, *Digital Image Processing,* Addison-Wesley, New York, 1993.

28. S. Lyu and H. Farid, "Steganalysis using color wavelet statistics and one-class support vector machines," *Security, Steganography, and Watermarking of Multimedia Contents VI*, E. J. Delp III and P. W. Wong, editors, *Proc. of SPIE*, **5306**, pp. 35-45, 2004.

29. Yun Q.Shi and Huifang Sun, *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*, CRC Press, 2000.