Analysis and Experiments on LSB-Based and ATD Steganographic Methods for Gray Scale and Color Images

Hajer Ahmed Alaswed

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
May 2017
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Mustafa Tümer
Director

I certify that this thesis satisfies the requirements as thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Alexander Chefranov
Co-Supervisor

Assoc. Prof. Dr. Gürcü Öz
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Alexander Chefranov
2. Assoc. Prof. Dr. Gürcü Öz
3. Asst. Prof. Dr. Adnan Acan
4. Asst. Prof. Dr. Yiltan Bitirim
5. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

The aim of this thesis is the analysis and experiments on steganographic methods for gray scale and color images and study of quality measures of the stego-images. For gray images, the Algorithm with Ternary Digits (ATD) and Least Significant Bits with Threshold (LSBT) are explained in detail. In the existing studies, for LSBT algorithm, some information is not provided such as how to set the values of threshold, T, and two moduli, mu, ml. Moreover, our analysis proves that LSBT encounters problems when the value of a pixel is close to the threshold value, T. In the study, LSBT problems are resolved by imposing constraints on the threshold and moduli values. Known results are displayed as Peak Signal to Noise Ratio (PSNR) for different values of embedding capacity, without presenting the formula for threshold; hence, the threshold is formalized herein as a function of embedding capacity.

In ATD, the main idea is embedding a secret message (in each pixel, embedding two ternary digits) as a ternary numbers. In LSBT, the embedding capacity of each pixel is determined by comparing the pixel value versus the threshold, T: if the pixel value is greater than or equal to T, then $floor(log_2 mu)$, otherwise $floor(log_2 ml)$ is used as embedding capacity. According to our analysis, PSNR of LSBDT is greater than that of ATD when the embedding capacity is less than or equal to 3 BPP.

For color images, LSB and ATD are implemented for different color combinations of 8 BPP embedding capacity. According to our experiments, PSNR of LSB for color images had fluctuations in different combinations with the same embedding capacity

8 BPP while the PSNR of ATD was stable in different combinations. However, the value of WMSNR (1/3, 1/3, 1/3) for LSB with different combinations looked invariant for different combinations when compared with other weights. For LSB algorithm when comparison between different metrics is made by deviation evaluation of the metrics, the best metric for LSB algorithm is found as WMSNR (1/3,1/3,1/3), with the minimal deviation value 0.211. And the maximal deviation is obtained for WPSNR (0.4, 0.243, 0.357) corresponding to the human eye color perception. Thus, human color perception-originated weights are not appropriate for the images assessment.

Keywords: Steganography, Algorithm with Ternary Digits (ATD), Least Significant Bit with Threshold Algorithm (LSBT), Gray Scale Image, Color Image, Embedding Capacity, Image Quality Metrics, Peak Signal-to-Noise Ratio (PSNR).

ÖZ

Bu tezin amacı, gri tonlamalı ve renkli görüntüler için stenografik yöntemler kullanarak analiz ve deneyler yapmak ve stego görüntülerinin kalite ölçümlerini incelemektir. Burada gri görüntüler, üç basamaklı algoritma (ATD) ve eşiğe sahip en az anlamlı bitler (LSBT) detaylı olarak açıklanmıştır. LSBT algoritmaları için mevcut araştırmalarda, eşik değeri T ve iki modül olan mu ve ml değerlerinin nasıl ayarlandığı gibi bazı bilgiler verilmemiştir. Ayrıca, analizimiz bir pikselin değeri eşik değerine (T) yakın olduğunda, LSBT'nin sorunlarla karşılaştığını kanıtlıyor. Çalışmadaki, LSBT problemleri, eşik ve modül değerlerine kısıtlamalar getirerek çözüldü. Bilinen sonuçlar, gömme kapasitesinin farklı değerleri kullanılarak eşik için formül sunmadan tepe sinyal-gürültü oranı (PSNR) olarak biçimlendirilmiştir. Bu çalışmada eşik, gömme kapasitesinin bir fonksiyonu olarak biçimlendirilir.

ATD'de temel fikir, üç rakamlı olarak gizli bir mesaj gömmektir (her pikselde, iki üçer sayı gömülüdür). LSBT'de her pikselin gömme kapasitesi, piksel değeri eşik (T) ile karşılaştırılarak belirlenir: Eğer piksel değeri T'den fazla veya eşit ise, gömme kapasitesi olarak $floor(log_2 mu)$, değilse $floor(log_2 ml)$, kullanılır. Bizim analizlerimize göre gömme kapasitesi 3 BPP'den daha az veya eşit ise, LSBDT'nin PSNR değeri ATD'den fazladır.

Renkli görüntülerde LSB ve ATD uygulanırken 8 BPP yerleştirme kapasitesinin farklı renk kombinasyonları kullanıldı. Deneyimlerimize dayanarak, renkli görüntüler için LSB'nin PSNR'si, aynı gömme kapasitesi 8 BPP olan farklı kombinasyonlarda dalgalanma göstermekteyken, ATD'nin PSNR'ı farklı kombinasyonlarda sabit kalmıştır. Bununla birlikte LSB için WMSNR değeri (1/3, 1/3, 1/3), kombinasyonları ile birlikte farklı kombinasyonlara baktığımızda, diğer ağırlıklara kıyasla farklı kombinasyonlar için değişmez görünüyordu. LSB algoritması için farklı metrikler arasındaki karşılaştırma, metrikler için sapma değerlendirmesiyle yapılması durumunda, LSB algoritması için minimum sapma değeri 0.211 olan en iyi metrik olarak WMSNR (1/3, 1/3, 1/3) elde edildi. Yapılan deney sonuçlarına göre en fazla sapma, insan göz rengi algısına karşılık gelen WPSNR(0.4, 0.243, 0.357) değerlerinde saptandı. Çıkan sonuçlardan, insan renk algılamasına dayalı ağırlıkların görüntü değerlendirmesi için uygun olmadığı saptandı.

Anahtar Kelimeler: Steganografi, Üç Basamaklı Algoritma (ATD), En Az Önemli Bit Eşik Algoritması (LSBT), Gri Ölçekli Görüntü, Renkli Görüntü, Gömme Kapasitesi, Görüntü Kaliteli Metrikleri, Tepe Sinyal –Gürültü Oranı(PSNR).

DEDICATION

To my beloved husband, **Ahmed**.

ACKNOWLEDGMENT

First of all, I thank Allah so much for giving me the strength and potential to finish my Master. Secondly, I would like to express my special appreciation and thanks to my beloved husband for his love and support throughout my life. Also, I would like to thank my supervisors Assoc. Prof. Dr. Alexander Chefranov, and Assoc. Prof. Dr. Gürcü Öz for their guidance and support throughout this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ATD	Algorithm with Ternary Digits
AC	Average Color
ALSBmax	Adaptive Least Significant Bit max
ALSBmin	Adaptive Least Significant Bit min
APSNR	Actual Peak Signal to Noise Ratio
BPP	Bit Per Pixel
C-TDH	Coded Ternary Data Hiding
DWT	Discrete Wavelet Transform
DCT	Discrete Cosine Transform
ETLSM	Extended Table Lookup Substitution
EC	Embedding Capacity
FFT	Fast Fourier Transform
HSV	(Hue, Saturation, Value)
IRMDR	Improved Rightmost Digit Replacement
IWT	Integer Wavelength Transform
LSB	Least Significant Bit
LSBT	Least Significant Bit with Threshold
LSBT-M	Least Significant Bit with Threshold Modified
LSBCT	Least Significant Bit with Constant Threshold
LSBDT	Least Significant Bit with Dynamic Threshold
MSE	Mean Square Error
MSNR	Mean Signal to Noise Ratio
PSNR	Peak Signal to Noise Ratio

PPM	Pixel Pair Matching
PBPVD	Parity Bit Pixel Value Differencing
RGB	(Red, Green, Blue)
SNR	Signal to Noise Ratio
SMVQ	Side Match Vector Quantization
TLSM	Table Lookup Substitution
TDH	Ternary Data Hiding
WPSNR	Weighted Peak Signal to Noise Ratio
WAPSNR	Weighted Actual Peak Signal to Noise Ratio
WMSNR	Weighted Mean Signal to Noise Ratio

Chapter 1

INTRODUCTION

Steganography is an art of embedding information in other media files for example text, image, etc, where the embedding information will not be discovered [2], There are basically two types of techniques in the Steganography: spatial domain and frequency domain. In the spatial domain, the actual values are impacted to embed the secret information while in frequency domain, the cover object is converted to the frequency domain by Discrete Wavelet Transform (DWT) and then the secret information is embedded [3].

In this thesis, we consider three steganographic schemes in the spatial domain, Algorithm with Ternary Digits (ATD) [28] [18], Least Significant Bit (LSB) [10] [27], and LSB with Threshold (LSBT) [26] [9] in the gray and the color scale images. In [28], in ATD, the secret message is divided to non-overlapping pixel blocks, then that it is converted to the ternary number with digits from {0, 1, 2}. In each pixel of cover image, two ternary digits are embedded by converting the pixel of the cover image to binary (8 bits) and dividing it to two sub-segments and embedding one ternary digit in each sub-segment. In the embedding phase, a secret digit is compared versus (the value of sub segment modulo 3); if the result is equal to the secret ternary digit then sub segment is not modified, otherwise it is increased or decreased by one.

LSBT [26], uses the threshold value, T, and two moduli numbers (modulus upper, mu, modulus lower, ml). In the embedding phase, the pixel value of cover image is compared with the threshold value T. If it is larger than or equal to the threshold, mu is used, otherwise ml is used. The number of bits embedded in each pixel is determined according to value, Ec, and the secret message is divided into blocks according to the value, Ec, where $1 \leq Ec \leq 8$; it gives better results when $Ec < 4$.

Experiments are conducted on ATD and LSBT with quality metrics Peak Signal to Noise Ratio (PSNR) [28], but they do not provide the proofs for the methods (LSBT and ATD). Also, sufficient information for their implementation in both algorithms (LSBT and ATD) such as how to select the value of the threshold T and the values of mu and ml is not provided. Therefore, in this thesis, we provide the proofs for the methods (LSBT and ATD), and we fix the recovered problem with LSBT when the value of the pixel of cover image is close to the threshold value T by imposing constraints on the threshold and moduli values. Also, we propose dynamic threshold modification, LSBDT, where threshold is defined according to the embedding capacity, measured in Bit Per Pixel (BPP), to have results compatible with those presented in [28]. Furthermore, for LSBT we determine the value of two moduli and extend ATD to work on the color scale images. We found that PSNR shows large variance for the same embedding capacity depending on the bits distribution across a pixel. Hence, we propose and implement new quality criteria MSNR and APSNR which show significantly lower variance and find the best of them.

For gray scale images, experiments are conducted with the cover images of the size

512×512 pixels and random secret messages; the size of secret messages starts from (512*512*2) bits and increases by 30000 bits in each iteration. In every iteration, the values of PSNR are recorded. The comparison between three methods were considered: ATD, LSBCT, LSBDT. When embedding up to 3 BPP the LSBDT achieves a high value of PSNR 43 dB whereas ATD achieves 39 dB. However, when increasing the embedding capacity more than 3 BPP the PSNR of LSBDT drops sharply; it was 39 dB in 3 BPP then it slumps to 35 dB in 3.2 BPP while PSNR of ATD decreases slightly until 37 dB.

For color images, experiments are conducted with the cover image of the size 512×512 pixels and the constant size secret message. In this case, we implement LSB with different embedding combinations, (224, 242, 422, 134, 143, 314, 341, 413, 431), each of them representing 8 bits embedding in each pixel, for example, combination 134 means embedding one bit in Red, three bits in Green, and four bits in Blue color component of a pixel. Also, we modify LSB to adaptive LSB (ALSBmax, ALSBmin) and we extend ATD to work with color images which is implemented for different combinations (112, 211, 121) as four ternary digits embedded in each pixel (8 bits in each pixel). For example, combination 112 means embedding one ternary digit in Red, one ternary digit in Green, and two ternary digits in Blue color component of a pixel. According to our results for color images, for the same embedding capacity in the different combinations, we get a fluctuating value of PSNR for LSB, therefore we have studied the quality metrics and improved PSNR by introducing new metrics, MSNR and APSNR. Also, we weighted PSNR, MSNR, and APSNR by three different groups of weighs (R_w =0.4, G_w =0.3, B_w =0.3), (R_w =0.4, G_w =0.243, B_w =0.357), and (R_w =1/3, G_w =1/3, B_w =1/3) where R_w, G_w, and B_w are weights of Red, Green, and Blue, respectively. We have tested new metrics, MSNR, APSNR, on the color and gray scale images.

The thesis research is organized as shown in Figure 1.1. Chapter 2 discusses related work and introduces LSB, ATD, and LSBT methods, and the problem definition. In Chapter 3, ATD and LSBT are analyzed, problems for LSBT are recovered by building counterexamples, and fixed. Design and Implementation of ATD and LSBT-based methods on gray scale images are shown in Chapter 4. Design and implementation of ATD-based and LSB-based methods on color images are shown in Chapter 5. Chapter 6 shows results of experiments on the above methods and compares them with the known experiments results. Chapter 7 concludes the thesis.

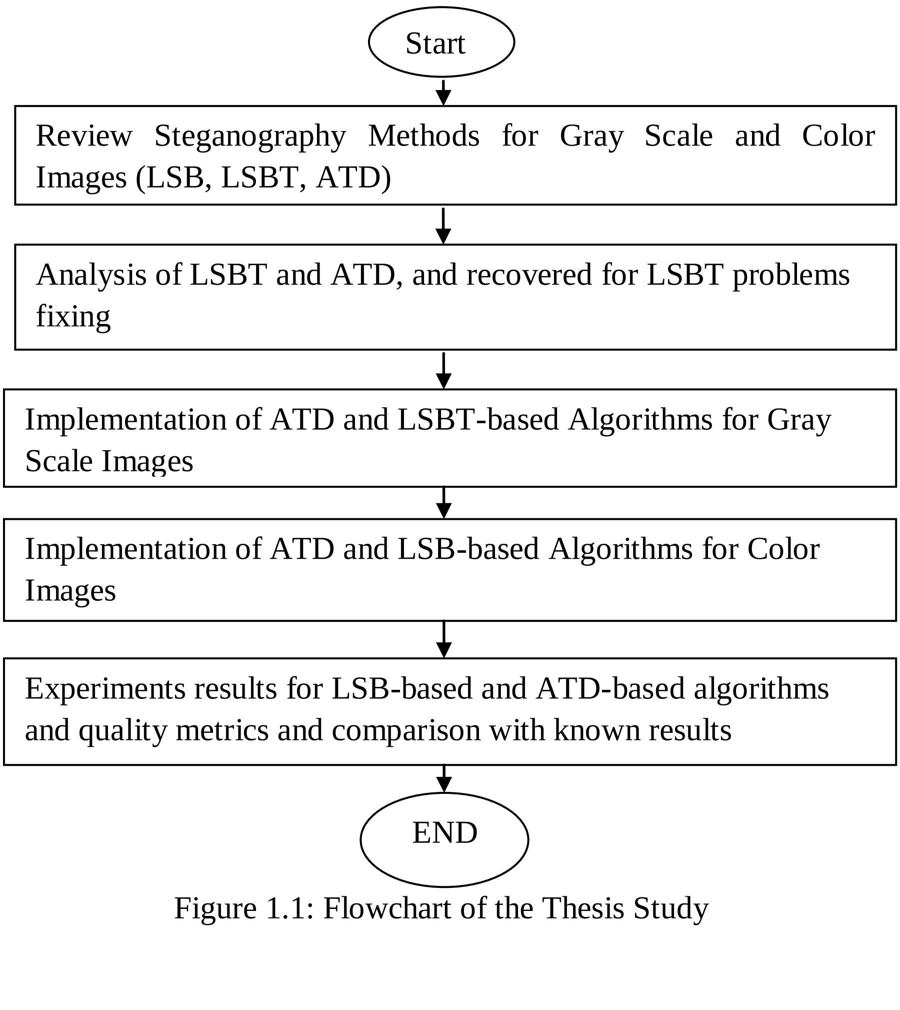


Figure 1.1: Flowchart of the Thesis Study

Chapter 2

RELATED WORK AND PROBLEM DEFINITION

2.1 Overview of Steganography

Nowadays, with the recent developments in electronic computer and its intrusion into our daily life, the need for private communication has increased. We use various techniques to provide secrecy in communication. One of such techniques is steganography; steganography is used to protect important data from unauthorized user by hiding secret data into other media files like text, image, etc. Steganography is a combination of two words, the first word "stegano" means the "cover", and the second word "graphic" means "writing", they are both Greek words [1].

2.2 Categories of Steganography

There are different categories of steganography; these categories can be split to five broad categories as the following [16]:

- Text Steganography

This is the most popular technique of steganography, it is done by embedding secret data in text, by modifying the spacing between the words and line [16].

- Image Steganography

In this technique, the secret data is embedded in an image. This image can be a color scale image, gray scale image or binary image. Color scale image has large space for embedding secret data therefore color scale image Steganography is more popular than gray scale image in Steganography. Color scale images can be represented in different formats such as RGB (Red, Green, Blue), HSV (Hue, Saturation, Value)

[3].

• Audio Steganography

Audio Steganography is hiding the secret data in an innocuous cover speech in a secure and robust manner. There are various ways popularly used in audio Steganography such as LSB coding, parity coding, and spread spectrum [16].

• Video Steganography

Video Steganography is a combination of image steganography and audio Steganography as the video consists of a set of images and audio [16].

• Protocol Steganography

In protocol steganography, the secret data is hidden in the header of a TCP/IP [16].

2.3 Methods of Steganography

There are two classes of methods of steganography techniques as that [3]:

1) Frequency Domain Methods.

2) Spatial Domain Methods.

2.3.1 Survey of Frequency Domain Methods

In transform domain, the cover object is converted to another domain to get the transformed coefficients, these coefficients are changed to hide the secret data. After that, these coefficients are transformed back to the spatial domain to get stego object. The widely used transforms are Discrete Wavelet Transform (DWT), Discrete Cosine Transform (DCT) and Fast Fourier Transform (FFT)[3].

In DCT is the core of image coding and video compression techniques. Such as for JPEG image format, image is divided into 8x8 pixel blocks then the DCT is applied to each block to get 64 DCT coefficients each [13]. The FFT is popularly used for frequency analysis which is easy to get the phase of a coefficient and change it by secret data. As in DCT an image is divided into several non-overlapped blocks [12]. There are a number of schemes working in the frequency domain they are as the follows:

In 2002 Chang, Chen, and Chung proposed a scheme for hiding the secret message in the cover image based upon JPEG and quantization table modification [7]. This method uses the JPEG image as a cover image. the cover-image divided into non-overlapping blocks of 8*8 pixels, and then applies DCT on each block to transform into DCT coefficients, the secret message embedded in the middle frequency part of the quantized DCT coefficients for each block. after embedding, use entropy coding to compress each block to obtain a JPEG file. This scheme achieves the larger capacity and the quality of the stego-images is acceptable

In 2013, Acharya, Hemeletha, Renuka, and Kamath proposed a method for hiding gray scale image in color scale image by utilizing Integer Wavelet Transform (IWT) and Discrete Wavelet Transform (DWT) [4]. In this method, the secret image is not embedding in a cover image, instead, generated the key by using DWT then this key is hiding in color scale image with run length encoded by using IWT. So, this method achieves the improving the security and has a high quality of the stego image compared to other methods.

In 2013 Gupta and Sharma proposed a scheme for hiding gray scale image in audio by utilizing Discrete Wavelet Transform (DWT) and Least Significant Bit (LSB) [14]. In this method embedding secret image bits in the higher frequency component of the audio file after applying the DWT on the audio, for hiding the audio is divided into blocks, the size of each block is 8*8 pixels and then store the image bits into the last 3 bits of the audio file.

In 2014 Chen, Zhang, Ma, and Yu proposed a method for reversible data hiding in DCT domain by recursive code construction [8]. In this method, the data is embedded over a special ternary cover that is suitable for any transform domain, for example, DCT domain. The likelihood density function of the transformed coefficients has a Laplacian distribution with a small variation. Thus, this technique has good image quality.

In 2014, Lavania, Matey, and Thanikaiselvan proposed a method for embedding secret data in the medical image (color scale image) for the real-time application by using the Integer Wavelength Transform (IWT) [21]. In this method, the data hiding in the red plane of cover image by dividing the cover image to non-overlapping pixel block the size of block 8*8 pixels and then apply IWT on each block and embedding L random steno bit message in LH, HL and HH coefficient of a block. So, this scheme achieved high capacity and robustness.

In 2014, Garg and Mathur proposed a method for embedding the secret gray image in a gray image by using fractional Fourier transform (FFT) and wavelet coefficients (DWT) [15]. In this method apply FFT on both images (secret image and cover image) which are divided into real and imaginary part then apply DWT on the real part of both images, the embedding process done by add approximation of cover image and secret image by using alpha blending. So, this method achieves both robustness and higher security.

In 2015, Acharya, Hemalatha, and Renuka proposed a method for embedding audio in color scale image by using the wavelet transform [3]. In this method, the cover image is presented in YCbCr format, and then Cb, Cr components and secret audio are transferred to wavelet domain by using IWT. The approximate coefficients of secret audio are embedded in a second and third bit of high-frequency coefficients of Cb and Cr. This method shows a high quality of the stego image.

2.3.2 Survey of Spatial Domain Methods

Spatial domain implies the real location of a pixel in media forms such as text, image, video, sound, etc. While hiding secret data in a pixel, the location of a pixel is considered and then the pixel value is used to hide the data. There are many steganography algorithms working on color or gray scale images; each algorithm has its own protection mechanism and complication techniques since the basic aim for all of these algorithms is encoding a large amount of secret data and little effect on the cover file. It means large value for the Bit Per Pixel (BPP) embedding capacity with high image quality.

- Survey of Steganography Methods for the Color Scale Images

There are many techniques of steganography working on color scale images as the follows:

In 2007, Yu, Chang and Lin proposed a scheme for embedding a color or a gray scale image in a true color scale image [29]. This scheme uses three different types of secret image: color scale image, a palette-based 256-color scale image, and a gray scale image. A secret image is converted to a binary representation, and then the secret data are protected by encrypting using DES algorithm. After that, each 8-bit byte of encrypted data is divided into 3 bits, 2 bits, and 3 bits. This method achieves high quality of the image.

In 2013, Kiruba and Karthikeyan proposed a method for detection of adaptive pixel pair matching in color scale images and gray scale images [19]. This technique is based on pixel pair matching (PPM) for data hiding. The basic idea of PPM is to utilize the values of pixel pairs as a reference assortment and seeking a coordinate in the neighborhood set of this pixel pair according to a given message digit. In this method, the maximum value of the embedding capacity of the payload is 1.161 BPP. So, this method achieves the best quality image with less distortion.

In 2013, Maj, Pal and Roy proposed a method by using Sudoku puzzle for embedding a secret message in the color scale image [22]. In this method, the cover image is divided into equal-sized blocks, the size of a block 64 bits. In every block embedding a character of the secret message in each three pixels. So, this scheme achieved more robustness with less computation.

In 2015, Singh and Singh proposed a method to improve LSB for hiding secret data in color scale image [24]. In this method, the secret message embedding in LSB of color scale image uses 2-2-4 LSB insertion: this technique embeds 2, 2, and 4 bits of secret data into 2 LSB of a red component, 2 LSB of green component, and 4 LSB of blue component, respectively. This method achieved a high quality of the image and high embedding capacity.

- Survey of Steganography Methods for Gray Scale Images

There are many techniques of steganography working on gray scale images, one of the easiest, fast and very publicly known technique of steganography is the Least Significant Bit (LSB). In this technique, least significant bit or bits of a pixel are replaced by the bits of the secret data to be hidden. LSB can change up to 4 least significant positions from a byte. Changing four bits of a pixel may cause distortion in an image due to a noticeable change in intensity of a cover object [2].

One of the most common techniques is the LSB substitution method; it is a simple method [10]. The main idea of LSB substitution is to change the least significant bits of the cover file with the bits of secret data. The secret information is divided into blocks with size M where $1 \leq M \leq 8$. Overall, this method can realize a good image quality when $M \leq 3$, however, for $4 \leq M \leq 8$, there is a sorely drop in the image quality.

To upgrade LSB substitution, many steganographic techniques were proposed. In 2001, Wang and Lin proposed a method that uses an optimal LSB substitution and genetic algorithm [27], to solve the problem of hiding data in the MLSB of the cover image by using genetic algorithm, when M is large in order to get better image quality and high embedding capacity.

In 2003, Chang, Hsiao and Chan proposed method to find optimal LSB substitution in image hiding by dynamic programming strategy [9], this method, is the same as the previous method [27] but this method uses the dynamic programming instead of the genetic algorithm in finding the optimal value for M. Also, this method consumes less computation time. This method achieves a good quality image and less computational time.

In 2003, Zhang and Ping proposed a scheme for the reliable detection of least significant bit (LSB) basic on statistical observations on difference image histograms [30]. This method uses gray scale images with size 512×512 pixels, the secret messages are embedded by using the random LSB replacement method with embedding ratios varying from 0 to 100% in 10% increments. The algorithm is more accurate than the other techniques when embedding large messages.

In 2006, Chang, Tai and Lin proposed a scheme for digitally compressed images based on side match vector quantization (SMVQ) [6]. In this method, the cover image is encoded using SMVQ; then the compressed image is created. The SMVQ-compressed cover image is divided into non-overlapping blocks then the secret data are embedded in the blocks. This method achieves a large size of secret data, visual quality and compression rate compare with other methods using SMVQ.

In 2012, Taur, Lin, lee and Tao proposed a method for hiding data in DNA sequences based on table lookup substitution (TLSM) [25]. TLSM is to enhance the performance of data embedding technique called the substitution. The Base-t TLSM encodes the secret message with radix t to fully use the substitution table. In extended TLSM (ETLSM) method, the number of elements of selectable substitution table was raised by taking additional letters into account. This method has good capacity and security.

In 2015, Jheng, Chen and Huang proposed a method for data hiding based on histogram medication over ternary computers [18]. They proposed two methods for data hiding Ternary Data Hiding (TDH) and Coded-Ternary Data Hiding (C-TDH). In the both methods, the secret data was ternary (NAF format). TDH method achieves higher peak signal to noise (PSNR) and C-TDH method achieves increased amount of the embedded secret data compared to TDH method.

In 2016, Hussain, Wahab, Ho, Javed and Jung proposed a scheme for data embedding using parity bit pixel value differencing (PBPVD) and improved rightmost digit replacement (IRMDR) [17]. In this method, the cover image divided into non-overlapping pixel blocks, and then calculate the PBPVD and the IRMDR in each block by calculating the difference between pixel values in blocks. If the difference value of the block exists in the L_0 level, then implement iRMDR; otherwise, PBPVD implement.

In this thesis, we investigated LSB, ATD, and LSBT in the spatial domain and detailed explained in the following:

2.3.3 LSB Method

One of simplest and fastest method is LSB. In this method, the binary secret message is divided into blocks with R bits, and in every pixel of the cover image hiding one block [10]. LSB method is shown below.

- LSB Embedding Algorithm

Input: Secret data as binary, *S*; cover image, $V\ [N, M]$; where *N* is the number of rows; and *M* is the number of columns; *R* is size of the block.

Output: Stego image, $Y\ [N, M]$.

Step 1: Binary data S divide into blocks S_i of *R* bits where $S_i \in \{0,1\}^R$.

Step 2: Hide S_i in pixel $v_i, i \in \{0,..,MN-1\}$.

$$y_i = v_i - (v_i mod 2^R) + S_i \tag{2.1}$$

Where v_i is the i^{th}cover pixel of *V*, y_i is the i^{th}stego pixel of *Y*, S_i is a decimal value of *R*-bits.

Step 3: End.

Figure 2.1 shows the flowchart of the LSB embedding algorithm.

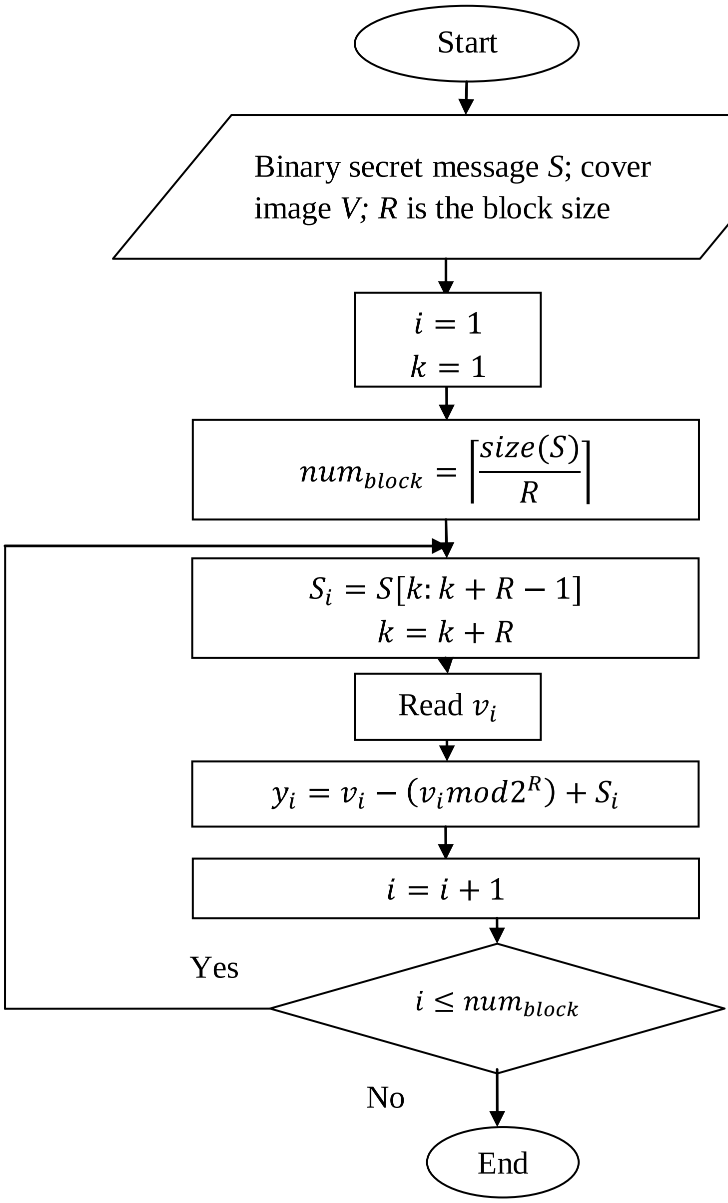


Figure 2.1: Flowchart of The LSB Embedding Algorithm

- LSB Extraction Algorithm

Input: Y is Stego image with size $[N, M]$; R is block size of the secret message.

Output: Secret data, S as binary.

Step 1: Set $S = \{ \quad \}$ empty set.

Step 2: Calculate the value of secret S_i for each pixel of stego image y_i at position i, $i \in \{0, .., MN - 1\}$ from next formula:

$$S_i = y_i mod 2^R \quad (2.2)$$

Where y_i is the stego pixel of *Y*.

Step 3: Transform every S_i into binary and insert S_i into the secret data *S*.

Step 4: End

Figure 2.2 shows the flowchart of the LSB extraction algorithm.

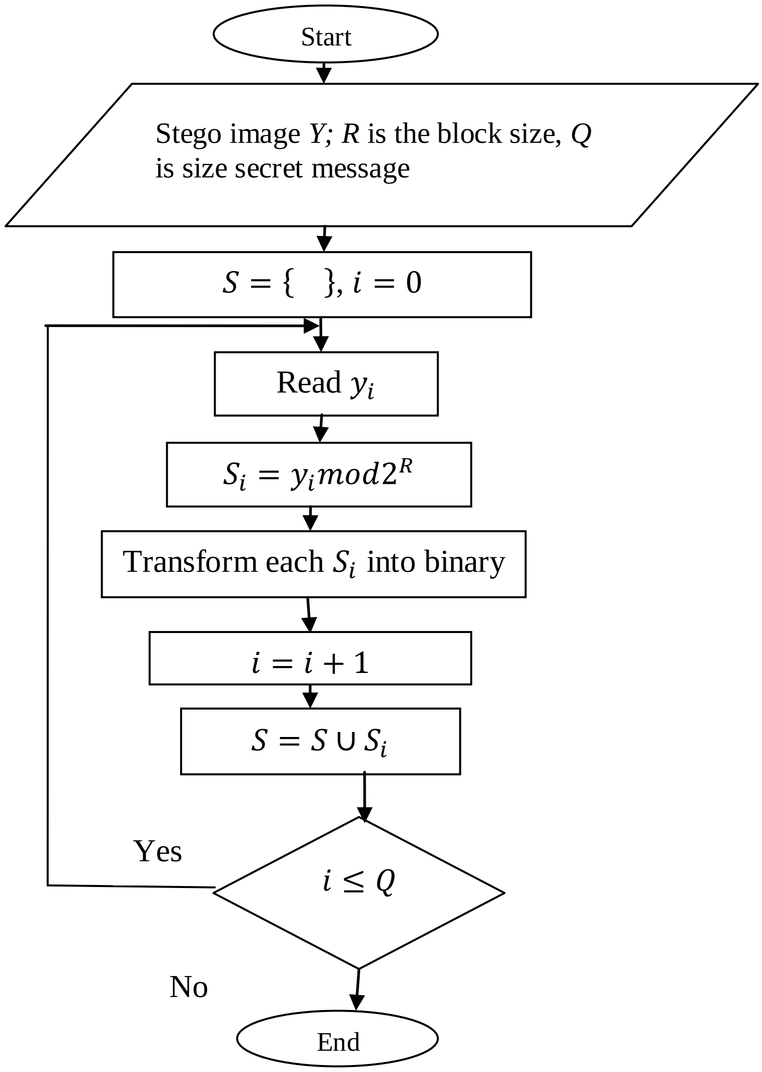


Figure 2.2: Flowchart of The LSB Extraction Algorithm

- Example 1. LSB Method

Let the pixel values of cover be (160 161 157 156), and the binary message be (11 01

10 01) and the size of block R=2 bits.

In block 1, we have $(11)_2 = (3)_{10} = S_1$ then

$$y_i = v_i - (v_i mod 2^R) + s_i$$

$$y_1 = 160 - (160\ mod\ 4) + 3 = 163$$

In block 2, we have $(01)_2 = (1)_{10} = S_2$ then

$y_2 = 161 - (161\ mod\ 4) + 1 = 161$.

In block 3, we have $(10)_2 = (2)_{10} = S_3$ then

$$y_3 = 157 - (157\ mod\ 4) + 2 = 158$$

In block 4, we have $(01)_2 = (1)_{10} = S_4$ then

$$y_4 = 156 - (156\ mod\ 4) + 1 = 157$$

In the extraction stage, if we have stego pixel values (163 161 158 157). For each stego pixel y_i, we get

$$s_i = y_i mod 2^R$$

$$s_1 = 163\ mod\ 4 = 3$$

$$s_2 = 161\ mod\ 4 = 1$$

$$s_3 = 158\ mod\ 4 = 2$$

$$s_4 = 157\ mod\ 4 = 1$$

Finally, we get $(3\ 1\ 2\ 1)_{10}$ and the binary secret message $(11\ 01\ 10\ 01)_2$ is restored.

2.3.4 Algorithm with Ternary Digits (ATD)

As proposed in [28], in this method the secret data consist of digits {0, 1, and 2} in the form of a ternary string. Each pixel of a cover image takes two ternary digits from the secret data. The ATD is shown below:

- ATD Embedding Algorithm

Input: Ternary secret message $S = \{S_k | 0 \le k \le |S|,\ S_k \in \{0,1,2\}\}$; V is cover image with size *[N, M]*, *N* is the number of rows; *M* is the number of columns. $V = \{v_{ij} | 0 \leq v_{ij} \leq 255$; v_{ij} is the i^{th}, j^{th} cover pixel.

Output: Stego image, v^{stego} $[N, M]$.

Step 0. $K = 0$

Step 1: Convert the pixel value $\boldsymbol{v_{ij}}$ to binary $b_7, b_6 \dots \dots b_0$ according to equation (2.3)

$$v_{ij} = \sum_{r=0}^{7} b_r * 2^r \quad (2.3)$$

Step 2: Divide $\boldsymbol{v_{ij}}$ into two sub-segments $sub1_{ij} = b_7, b_6, b_5, b_4, b_3, b_2$; and $sub2_{ij} = b_1, b_0$.

Step 3: Check overflow/underflow for $sub1_{ij}, sub2_{ij}$ according to (2.4), (2.5)

$$sub1\grave{}_{ij} = \begin{cases} 000001 & , if\ sub1_{ij} = 000000 \\ 111110 & , if\ sub1_{ij} = 111111 \\ sub1_{ij} & Otherwise \end{cases} \quad (2.4)$$

$$sub2\grave{}_{ij} = \begin{cases} 01 & , if sub2_{ij} = 00 \\ 11 & , if sub2_{ij} = 11 \\ sub2_{ij} & Otherwise \end{cases} \quad (2.5)$$

Step 4: Embed the first ternary secret number S_k in $sub1\grave{}_{ij}$ according to the next cases:

Case 1: $mod\left(sub1\grave{}_{ij}, 3\right) = S_k$

$$sub1_{ij}^{stego} = sub1\grave{}_{ij} \quad (2.6)$$

Case 2: $mod\left(sub1\grave{}_{ij} + 1,3\right) = S_k$

$$sub1_{ij}^{stego} = sub1\grave{}_{ij} + 1 \quad (2.7)$$

Case 3: $mod\left(sub1\grave{}_{ij} - 1,3\right) = S_k$

$$sub1_{ij}^{stego} = sub1\grave{}_{ij} - 1 \quad (2.8)$$

Step 5: Construct $\boldsymbol{v\grave{}_{ij}}$ by using equation (2.9)

$$v\grave{}_{ij} = sub1_{ij}^{stego} * 2^2 + sub2\grave{}_{ij} \quad (2.9)$$

Step 6: $If\ (k = k + 1) < |S|$ go to **Step 7** else go to **Step9.**

Step 7: Embed the second ternary secret number S_k into $v\grave{}_{ij}$, according to next cases:

Case 1: $mod(v\grave{}_{ij}, 3) = S_k$

$$v_{\text{ij}}^{stego} = v\grave{}_{ij} \quad (2.10)$$

Case 2: $mod(v\grave{}_{ij} + 1,3) = S_k$

$$v_{\text{ij}}^{stego} = v\grave{}_{ij} + 1 \quad (2.11)$$

Case 3: $mod(v\grave{}_{ij} - 1,3) = S_k$

$$v_{ij}^{stego} = v\grave{}_{ij} - 1 \quad (2.12)$$

Step 8: $k = k + 1,\ if\ k < |S|$, go to **Step1**

Step 9: End.

Note 1. Note that due to (2.5), embedding of the second digit by (2.10)-(2.12) does not change its *sub1* part, i.e. $sub1(\text{v}_{\text{ij}}^{stego}) = sub1\ (v\grave{}_{ij}) = sub1_{ij}^{stego}$.

Figure 2.3 shows the flowchart of the embedding algorithm.

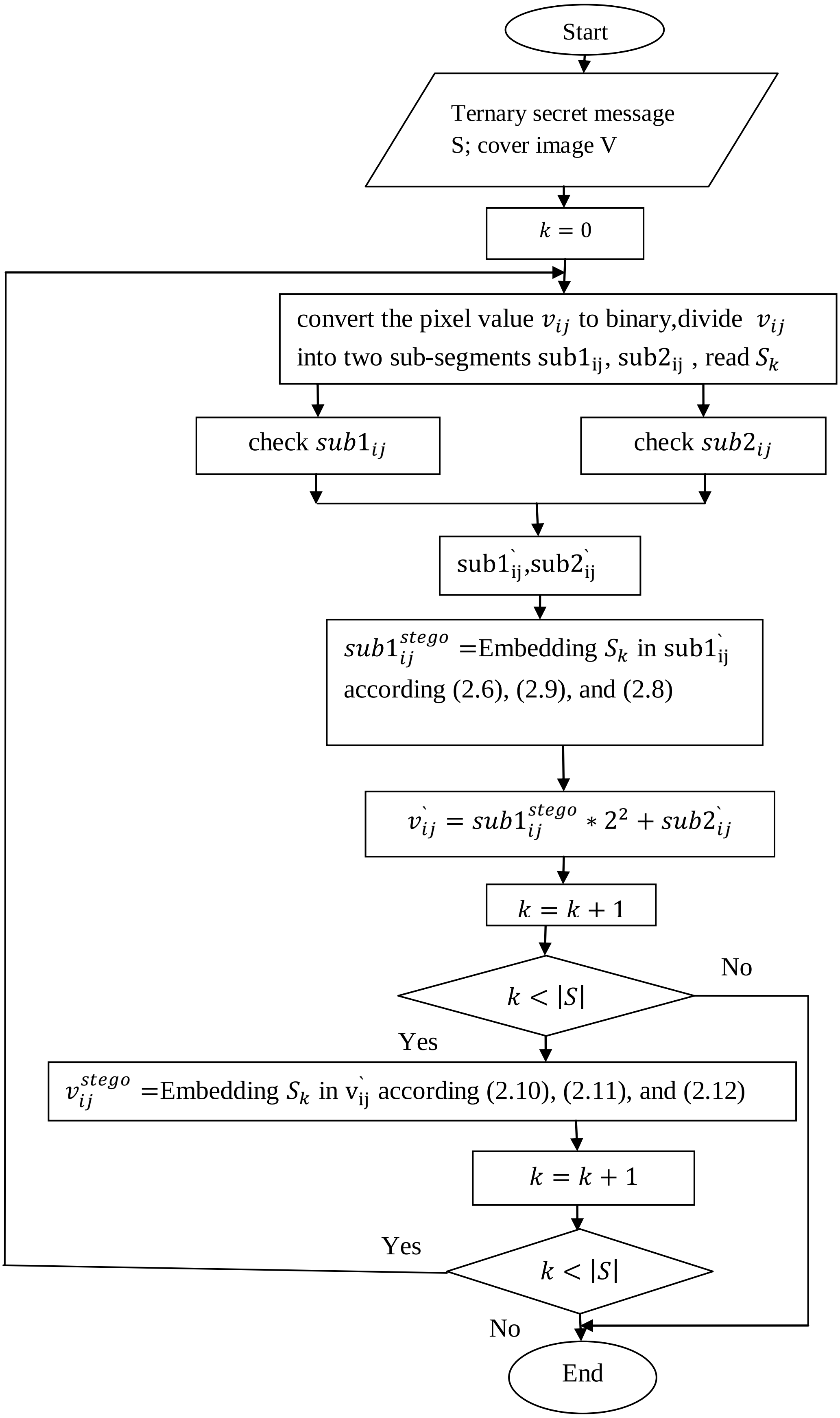


Figure 2.3: Flowchart of The ATD Embedding Algorithm

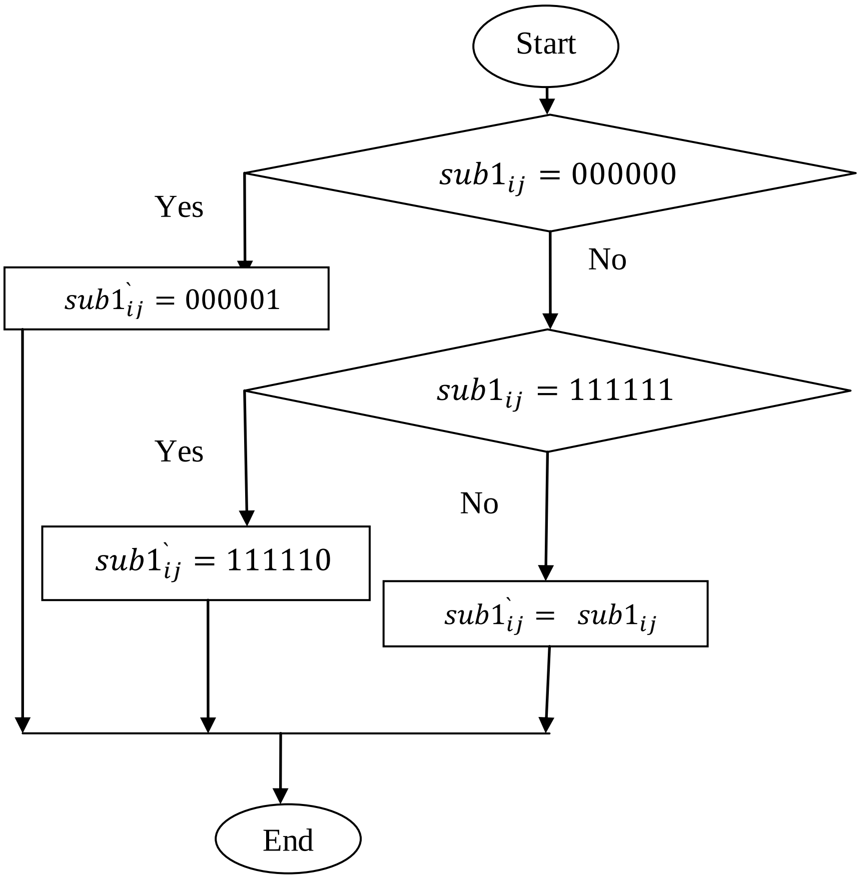


Figure 2.4: Flowchart of Check $sub1_{ij}$ Part of The ATD Embedding Algorithm

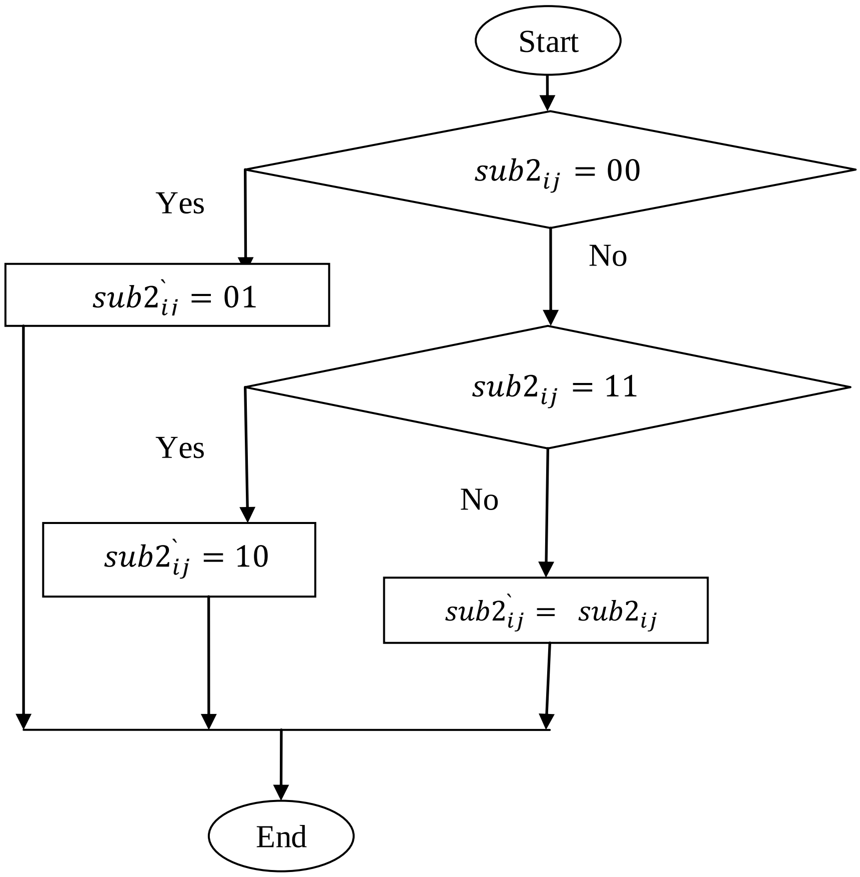


Figure 2.5: Flowchart of Check $sub2_{ij}$ Part of The ATD Embedding Algorithm

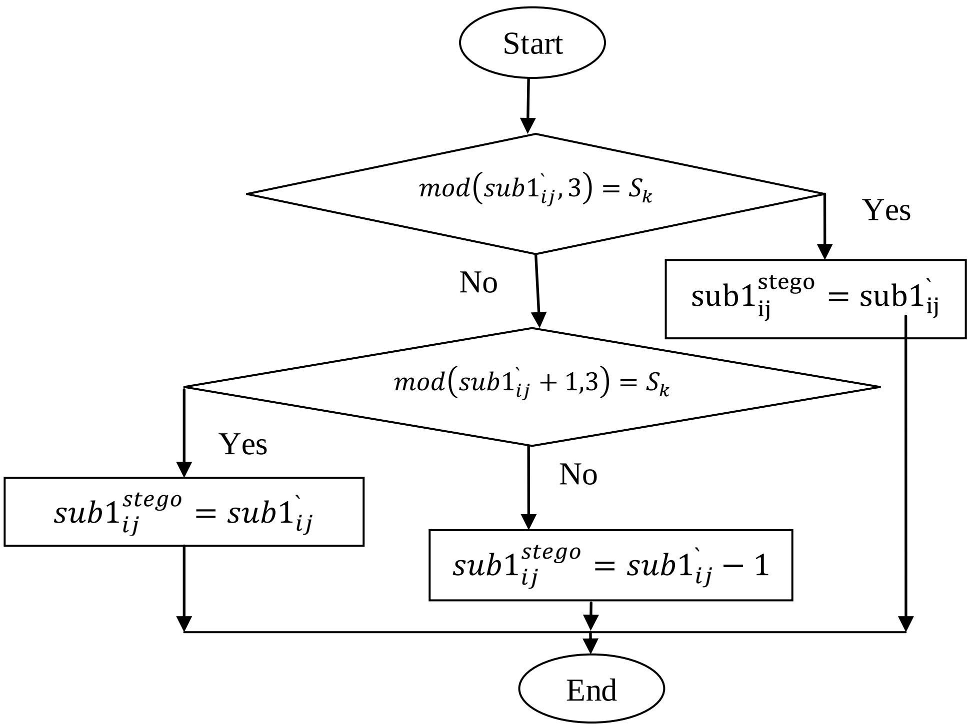


Figure 2.6: Flowchart of Embedding S_k in $sub1`_{ij}$ Part of The ATD Embedding Algorithm

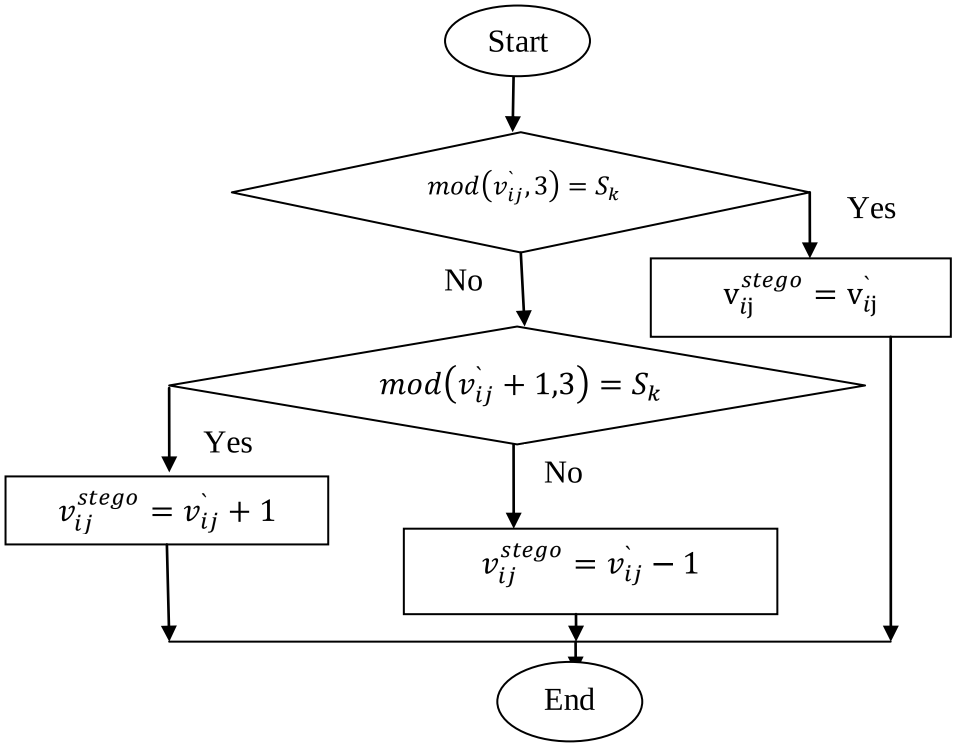


Figure 2.7: Flowchart of Embedding S_k in $v^{`}_{ij}$ Part of The ATD Embedding Algorithm

- ATD Extraction Algorithm

Input: *V* is stego image with size *[N, M]*, *N* is the number of rows; *M* is the number of columns. $V = \{v_{ij} | 0 \le v_{ij} \le 255\}$; v_{ij} is the i^{th}, j^{th} cover pixel.

Output: Ternary secret message, S.

Step 0. $k = 0$

Step 1: Convert the pixel value v_{ij} to binary $b_7 b_6 \ldots \ldots . . b_0$ according to equation (2.3)

Step 2: Divide v_{ij} into two sub-segments $sub1_{ij} = b_7 b_6 b_5 b_4 b_3 b_2$; and $sub2_{ij} = b_1 b_0$.

Step 3: Extract first ternary number of $sub1_{ij}$, according equation (2.13)

$$S_k = mod\left(sub1_{ij}, 3\right) \tag{2.13}$$

Step 4: $k = k + 1$, if $k < |S|$ go to **Step 5** else go to **Step 7**.

Step 5: Extract second ternary number from $\boldsymbol{v_{ij}}$, according toequation (2.14)

$$S_k = mod(v_{ij}, 3) \quad (2.14)$$

Step 6: $k = k + 1$, if $k < |S|$ go to **Step 1.**

Step 7: End.

Figure 2.8 shows flowchart of the extraction algorithm.

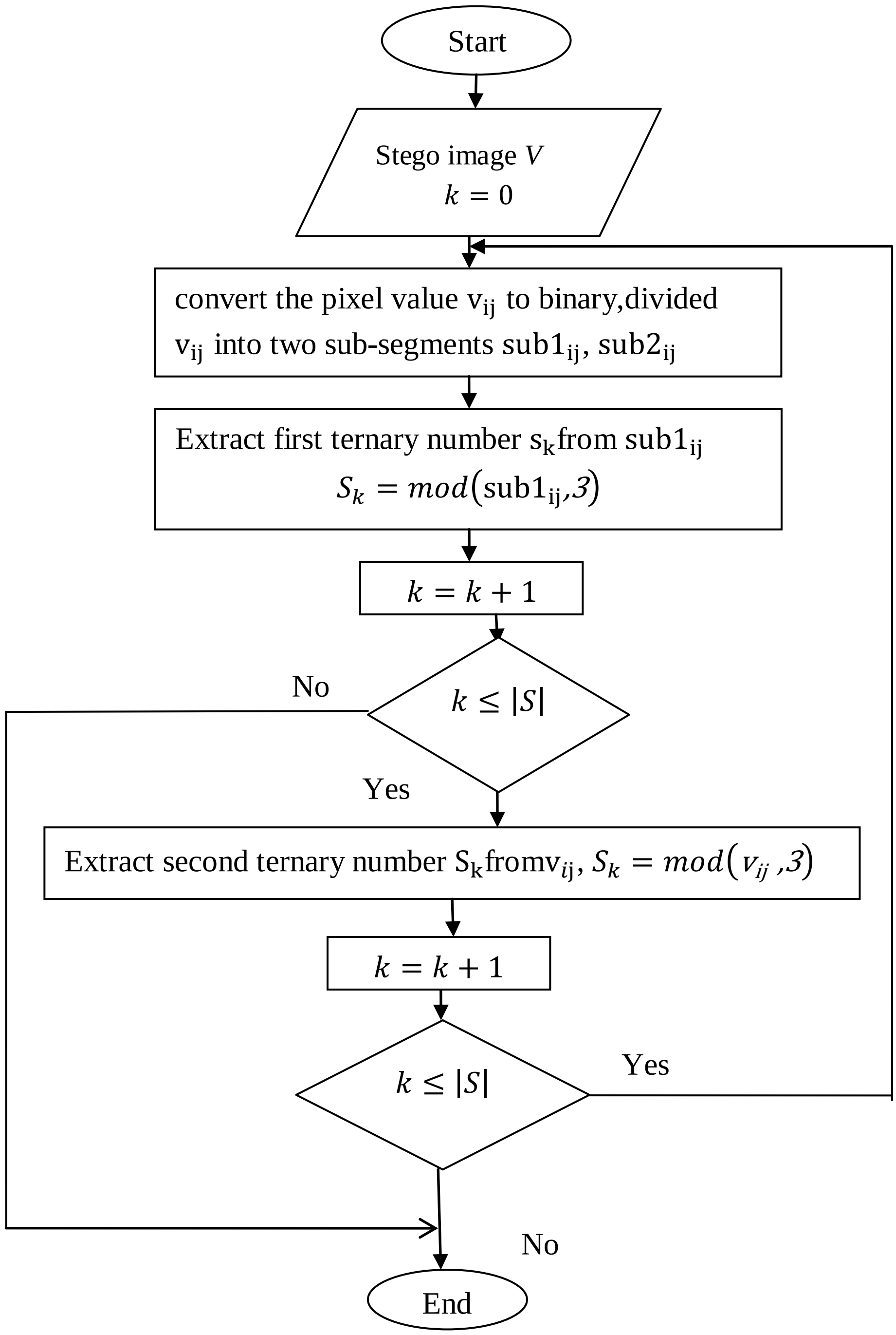


Figure 2.8: Flowchart of ATD Extraction Algorithm

• Example 2. Example for ATD Embedding and Extraction

Let the cover pixel values are $(135\ 137\ 138)_{10}$, and the ternary message be $(010212)_3$.We will embed two ternary digits into each cover pixel: 01 into 135, 02 into 157, and 12 into 138.

Embed 01 into cover pixel $v_1 = 135$.

Step 1: Convert cover pixel to binary

$$v_1 = (135)_{10} = (10000111)_2$$

Step 2: Divide binary value of cover pixel into sub_1, sub_2

$$sub_1 = (100001)_2 = (33)_{10}$$

$$sub_2 = (11)_2 = (3)_{10}$$

Step 3: Check overflow/underflow for sub_1, sub_2; the result after Check overflow/underflow according to (2.4) and (2.5):

$$sub_1 = (33)_{10}$$

$$sub_2 = (01)_2 = (2)_{10}$$

Step 4: Embed the first ternary secret number $S_1 = (0)_3$ to sub_1 according to (2.6), (2.7) and (2.8):

$$sub_1\ mod3{=}33\ mod\ 3{=}0\ {=}S_1$$

$$sub_1^{stego} = 33$$

Hence, apply (2.6)

Step 5: $\grave{v_1} = 33 * 2^2 + 2 = 134$.

Step 6: Read the second ternary number $S_2 = (1)_3$ and embed it in $\grave{v_1}$according to (2.10), (2.11) and (2.12):

$$v_1\ mod3 = 134\ mod\ 3 = 2\ \neq S_2$$

$$(v_1\ + 1)mod3 = (134 + 1)mod\ 3 = 0\ \neq S_2$$

$$(v_1\ - 1)mod3 = (134 - 1)mod3 = 133mod3 = 1 = S2$$

Hence, v_1 stego=133

Embed $(02)_3$ into cover pixel $\boldsymbol{v_2}$=137.

Step 1: Convert cover pixel to binary

$$v_2 = (137)_{10} = (10001001)_2$$

Step 2: Divide binary value of cover pixel into sub_1, sub_2

$$sub_1 = (100010)_2 = (34)_{10}$$

$$sub_2 = (01)_2 = (1)_{10}$$

Step 3: Check overflow/underflow for sub_1, sub_2 according to (2.4) and (2.5); no need for change in this step.

Step 4: Embed $S_3 = (0)_3$ into sub_1 according to (2.6), (2.7) and (2.8):

$$sub_1 \text{ mod3} = 34\, mod\; 3 = 1 \neq S_3$$

$$sub_1 \text{ mod3} = (34 + 1)\; mod\; 3 = 2 \neq S_3$$

$$sub_1 \text{ mod3} = (34 - 1)\text{mod3} = 33\text{mod3} = 0 = \text{S3}.$$

Hence, apply (2.6): sub_1 stego=33

Step 5: $\grave{v_2} = 33 * 2^2 + 1 = 133.$

Step 6: Read the next ternary number $S_4 = (2)_3$ and embed it in $\grave{v_2}$according to (2.10), (2.11) and (2.12):

$$v_2 \text{ mod3} = 133\; mod\; 3 = 1 \; \neq S_4$$

$$v_2 \text{ mod3} = (133 + 1) mod\; 3 = 2 = S_4$$

Hence,

$v_2^{stego} = 134$.

Embed $(12)_3$ into cover pixel v_3=138.

Step 1: Convert cover pixel to binary

$$v_3 = (138)_{10} = (10001010)_2$$

Step 2: Divide binary value of cover pixel into sub_1, sub_2

$$sub_1 = (100010)_2 = (34)_{10}$$

$$sub_2 = (10)_2 = (2)_{10}$$

Step 3: Check overflow/underflow for $sub_1,\ sub_2$ according to (2.4) and (2.5); no need for changes in this step.

Step 4: Embed $S_5 = (1)_3$ into sub_1according to (2.6), (2.7) and (2.8):

$$sub_1 \text{ mod3} = 34\, mod\ 3 = 1 = \ S_5$$

Hence, apply (2.6):

$$sub_1^{stego} = 34.$$

Step 5: $v_3^{`} = 34 * 2^2 + 1 = 137.$

Step 6: Read the next ternary number $S_6 = (2)_3$ and embed it in $v_3^{`}$ according to (2.10), (2.11) and (2.12):

$$v_3^{stego} = 137\, mod\ 3 = 2\ = S_6$$

$$v_3^{stego} = 137$$

The stego pixels are (133, 134, and 137).

In the extraction, if we have stego pixel value (133, 134, 137)

stego pixel v_1=133.

Step 1: Convert cover pixel to binary

$$v_1 = (133)_{10} = (10000101)_2$$

Step 2: Divide the binary value of the cover pixel into sub_1, sub_2

$$sub_1 = (100001)_2 = (33)_{10}$$

$$sub_2 = (01)_2 = (1)_{10}$$

Step 3: Extract first ternary number S_1 from sub_1 according to (2.13)

$$S_1 = 33\, mod\ 3 = 0$$

Step 4: Extract second ternary number S_2 fromv_1, according to (2.14),

$S_2 = 133\ mod\ 3 = 1$

The first part of the secret message $(01)_3$ is restored.

Next stego pixelv_2=134.

Step 1: Convert cover pixel to binary

$$v_2 = (134)_{10} = (10000110)_2$$

Step 2: Divide the binary value of cover pixel into sub_1, sub_2

$$sub_1 = (100001)_2 = (33)_{10}$$

$$sub_2 = (10)_2 = (2)_{10}$$

Step 3: Extract S_3 from sub_1 according to (2.13)

$$S_3 = 33\ mod\ 3 = 0$$

Step 4: Extract S_4 From v_2 according to (2.14)

$$S_4 = 134\ mod\ 3 = 2$$

The second part of the secret message $(02)_3$is restored.

Next stego pixel v_3=137.

Step 1: Convert cover pixel to binary

$$v_3 = (137)_{10} = (10001001)_2$$

Step 2: Divide the binary value of the cover pixel into sub_1, sub_2

$$sub_1 = (100010)_2 = (34)_{10}$$

$$sub_2 = (01)_2 = (1)_{10}$$

Step 3: Extract S_5 From sub_1 according to (2.13)

$$S_5 = 34\ mod\ 3 = 1$$

Step 4: Extract S_6 From v_3 according to (2.14)

$$S_6 = 137\ mod\ 3 = 2$$

The third part of the secret message $(12)_3$ is restored.

Finally, we embedded $(010212)_3$ into $(135,137,138)_{10}$ and the ternary secret message $(010212)_3$ is restored from the stego pixels$(133,134,137)_{10}$.

2.3.5 LSB with Threshold Algorithm (LSBT)

As proposed in [26], this method uses a threshold value which indicates the number of bits of the secret data embedded into the cover image. The LSBT algorithm is presented below:

• LSBT Embedding Algorithm

Input: B_s is a secret message as a bit string, two moduli numbers, mu and ml; P as cover image with size *[N, M]*, M is the number of rows; N is the number of columns; $k = 0$; T is the threshold value; $P = \{p_{ij} | 0 \leq p_{ij} \leq 255\}$; p_{ij} is the i^{th}, j^{th}cover pixel.

Output: Stego image, P_s.

Step 0: k=0

Step 1: $If\ p_{ij} \geq T$

$$EC = \lfloor \log_2 mu \rfloor, \quad (2.15)$$

$$RES = p_{ij}\ mod\ mu, \quad (2.16)$$

Else

$$EC = \lfloor \log_2 ml \rfloor, \quad (2.17)$$

$$RES = p_{ij}\ mod\ ml, \quad (2.18)$$

where $\lfloor x \rfloor$ is the maximal integer less or equal to x.

Step 2: Compute

$$D = |RES - DEC|, \quad (2.19)$$

where DEC is the decimal value of the next EC bits from B_s. From (2.15), (2.16),

Step 3: Embed DEC into p_{ij}. $k = k + EC$.

Case 1: $p_{ij} < T$

Case 1.1: $p_{ij} < \frac{ml}{2}$ (2.20)

$$P_{sij} = DEC \tag{2.21}$$

Case 1.2 $\frac{ml}{2} \leq p_{ij} < T - \frac{ml}{2}$ (2.22)

Case 1.2.1 $D > \frac{ml}{2}$ (2.23)

$$AV = ml - D. \tag{2.24}$$

Case 1.2.1.1 $RES \geq DEC$ (2.25)

$$P_{sij} = p_{ij} + AV. \tag{2.26}$$

Case 1.2.1.2 $RES < DEC$ (2.27)

$$P_{sij} = p_{ij} - AV. \tag{2.28}$$

Case 1.2.2 $D \leq \frac{ml}{2}$ (2.29)

$$AV = D. \tag{2.30}$$

Case 1.2.2.1 $RES \geq DEC$ (2.31)

$$P_{sij} = p_{ij} - AV. \tag{2.32}$$

Case 1.2.2.2 RES<DEC (2.33)

$$P_{sij} = p_{ij} + AV. \tag{2.34}$$

Case 1.3 $\left(T - \frac{ml}{2}\right) \leq p_{ij} < T$ (2.35)

$$P_{sij} = p_{ij} - RES + DEC. \tag{2.36}$$

Case 2:$p_{ij} \geq T$

$$k = k + EC.$$

Case 2.1 $p_{ij} > 255 - mu/2 + 1$ (2.37)

$$P_{sij} = 255 - mu + 1 + \text{DEC}. \tag{2.38}$$

Case 2.2$\left(T + \frac{mu}{2}\right) < p_{ij} \leq (255 - \frac{mu}{2} + 1)$ (2.39)

Case 2.2.1 $D > \frac{mu}{2}$ (2.40)

$$AV = mu - D \tag{2.41}$$

Case 2.2.1.1 $RES \geq DEC$ (2.42)

$$P_{sij} = p_{ij} + AV \quad (2.43)$$

Case 2.2.1.2$RES < DEC$ (2.44)

$$P_{sij} = p_{ij} - AV \quad (2.45)$$

Case 2.2.2 $D \leq \frac{mu}{2}$ (2.46)

$$AV = D \quad (2.47)$$

Case 2.2.2.1 $RES \geq DEC$ (2.48)

$$P_{sij} = p_{ij} - AV \quad (2.49)$$

Case 2.2.2.2. RES<DEC (2.50)

$$P_{sij} = p_{ij} + AV \quad (2.51)$$

Case 2.3 $T \leq p_{ij} < T + \frac{mu}{2}$ (2.52)

$$P_{sij} = p_{ij} - RES + DEC \quad (2.53)$$

Step 4: If $k < |B_s|$ go to **Step 1**.

Step 5: End.

Figure 2.9 shows the flowchart of the LSBT embedding algorithm.

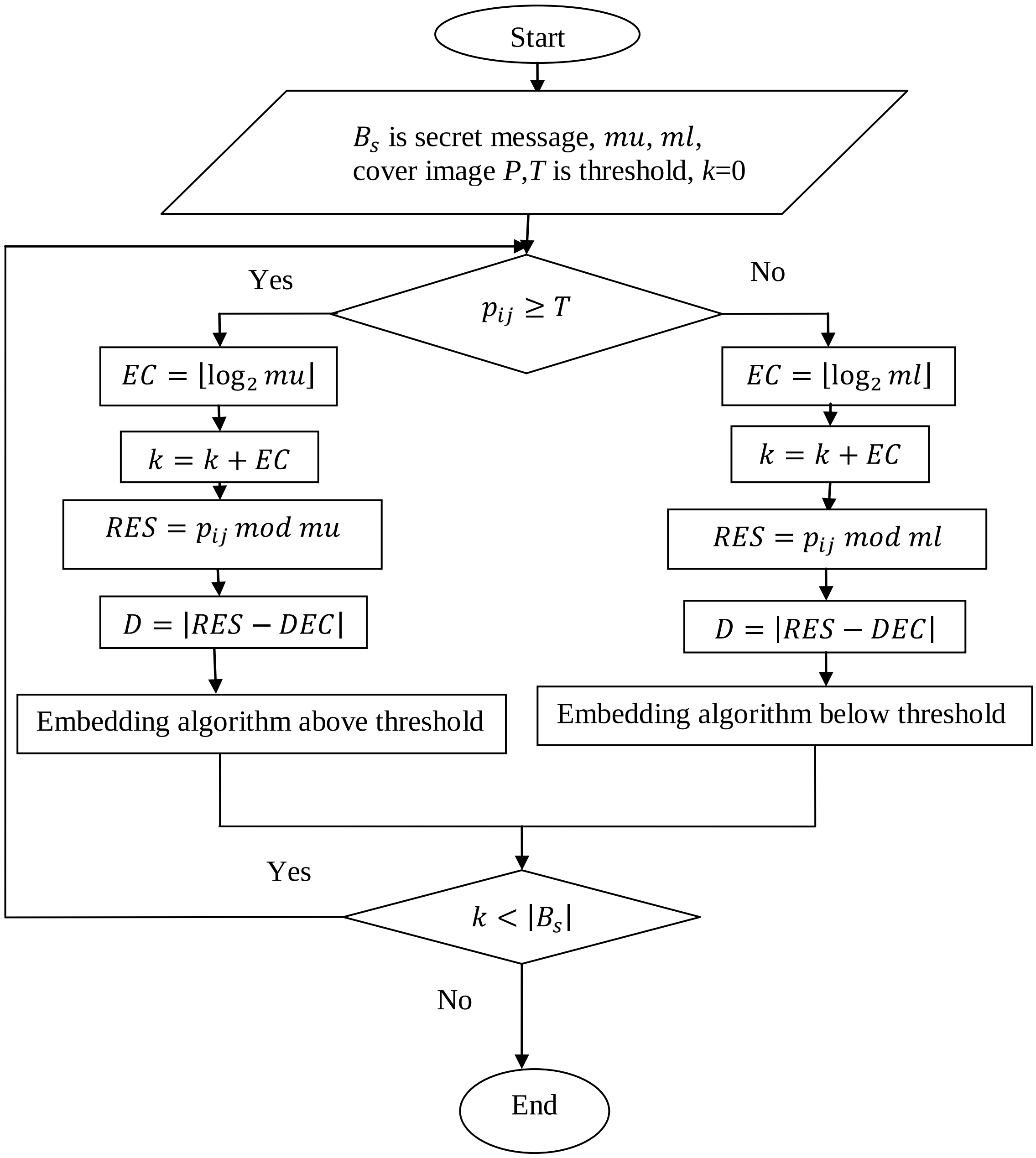


Figure 2.9: Flowchart for LSBT Embedding Algorithm

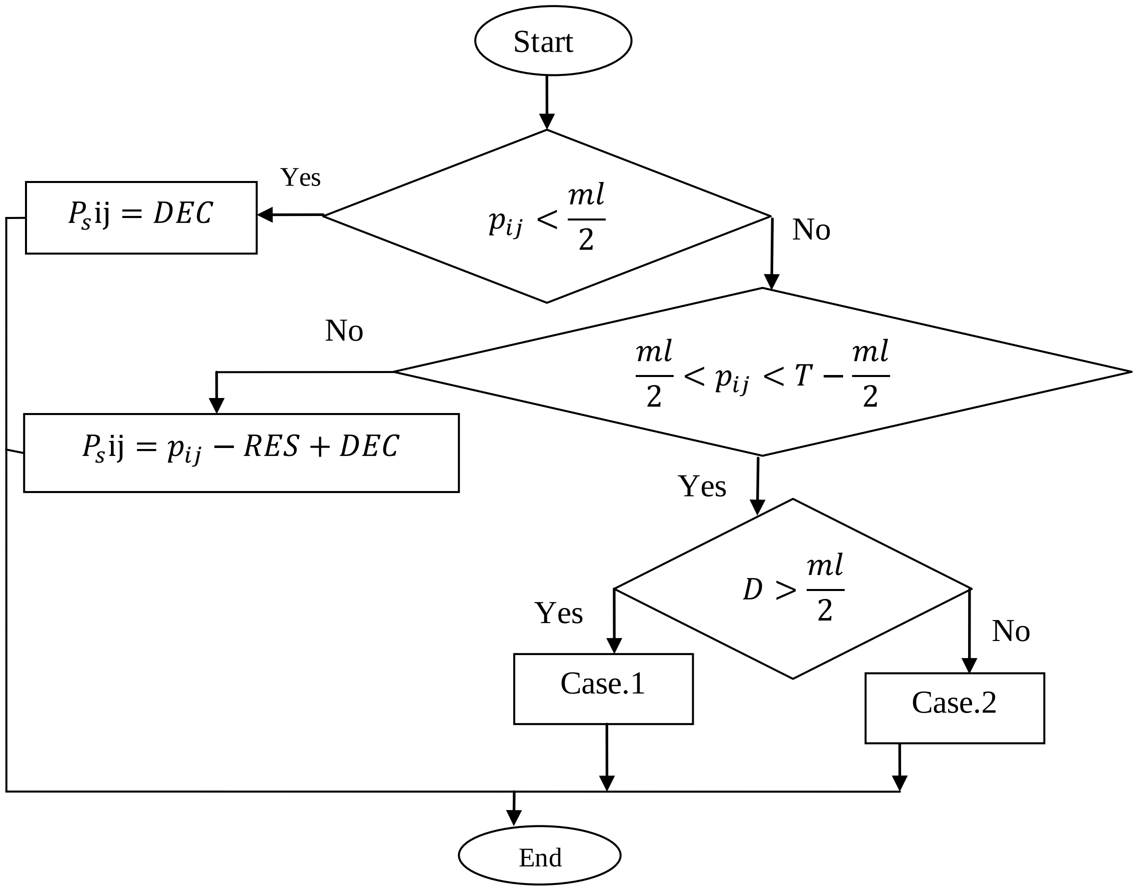


Figure 2.10: Flowchart of Embedding Algorithm Below Threshold Part of LSBT

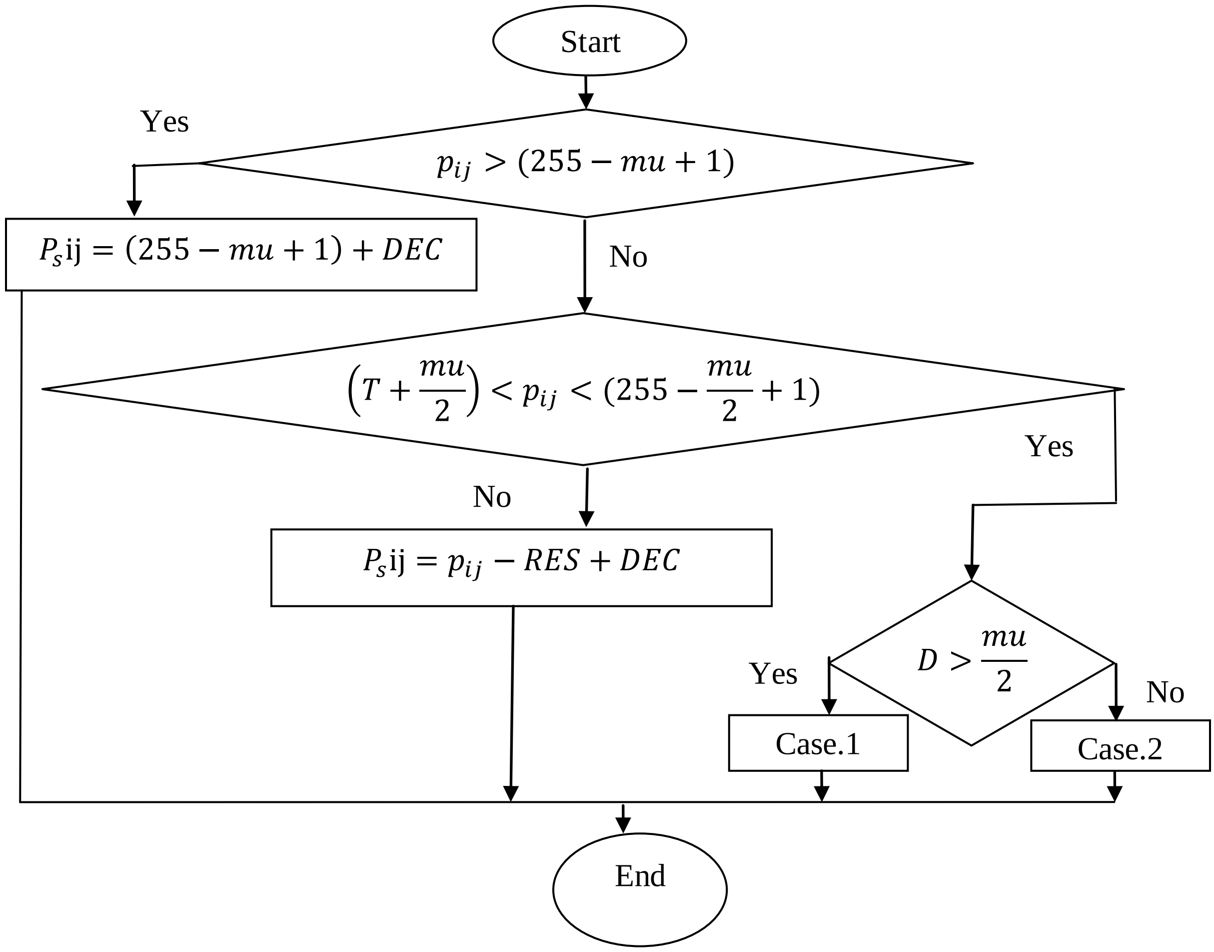


Figure 2.11: Flowchart of Embedding Algorithm Above Threshold of LSBT

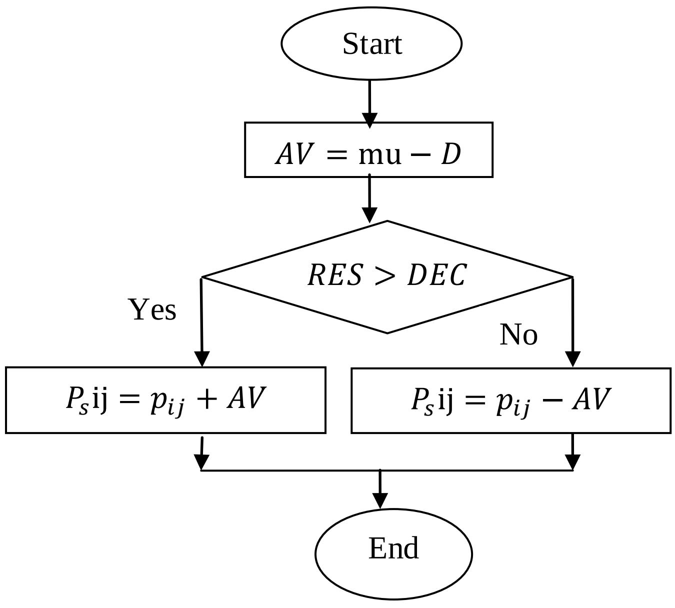


Figure 2.12: Flowchart of Case.1 in Figures 2.10, 2.11

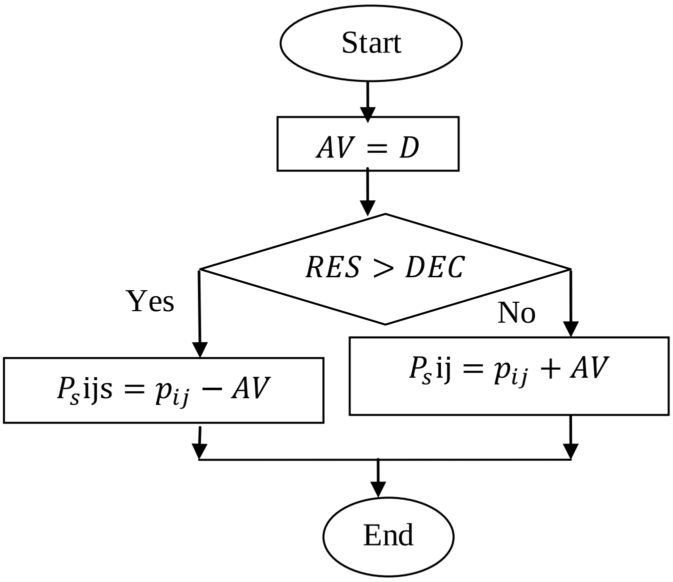


Figure 2.13: Flowchart of Case.2 in Figures 2.10, 2.11

• LSBT Extraction Algorithm

Input: P_s is stego image with size [N,M], T Threshold value, Two moduli, mu, ml.

Output: B_s is a bit string with extracted secret message

Step 1: *k=0*; Compute *RES* and *EC* as follows:

Case 1:$P_{sij} < T$

$$EC = \lfloor \log_2 ml \rfloor \tag{2.54}$$

$$RES = P_{sij} \; mod \; ml \tag{2.55}$$

$$k = k + EC$$

Case 2: $P_{sij} \geq T$

$$EC = \lfloor \log_2 mu \rfloor \tag{2.56}$$

$$RES = P_{sij} modmu \tag{2.57}$$

$$k = k + EC$$

Step 2: Convert RES into the bit string with the EC bits and insert it into the secret data B_s.

Step 3: if $k < |B_s|$ go to **Step** 1. Otherwise, go to **Step 4**.

Step 4: End

Figure 2.14 shows the flowchart of the LSBT extraction algorithm

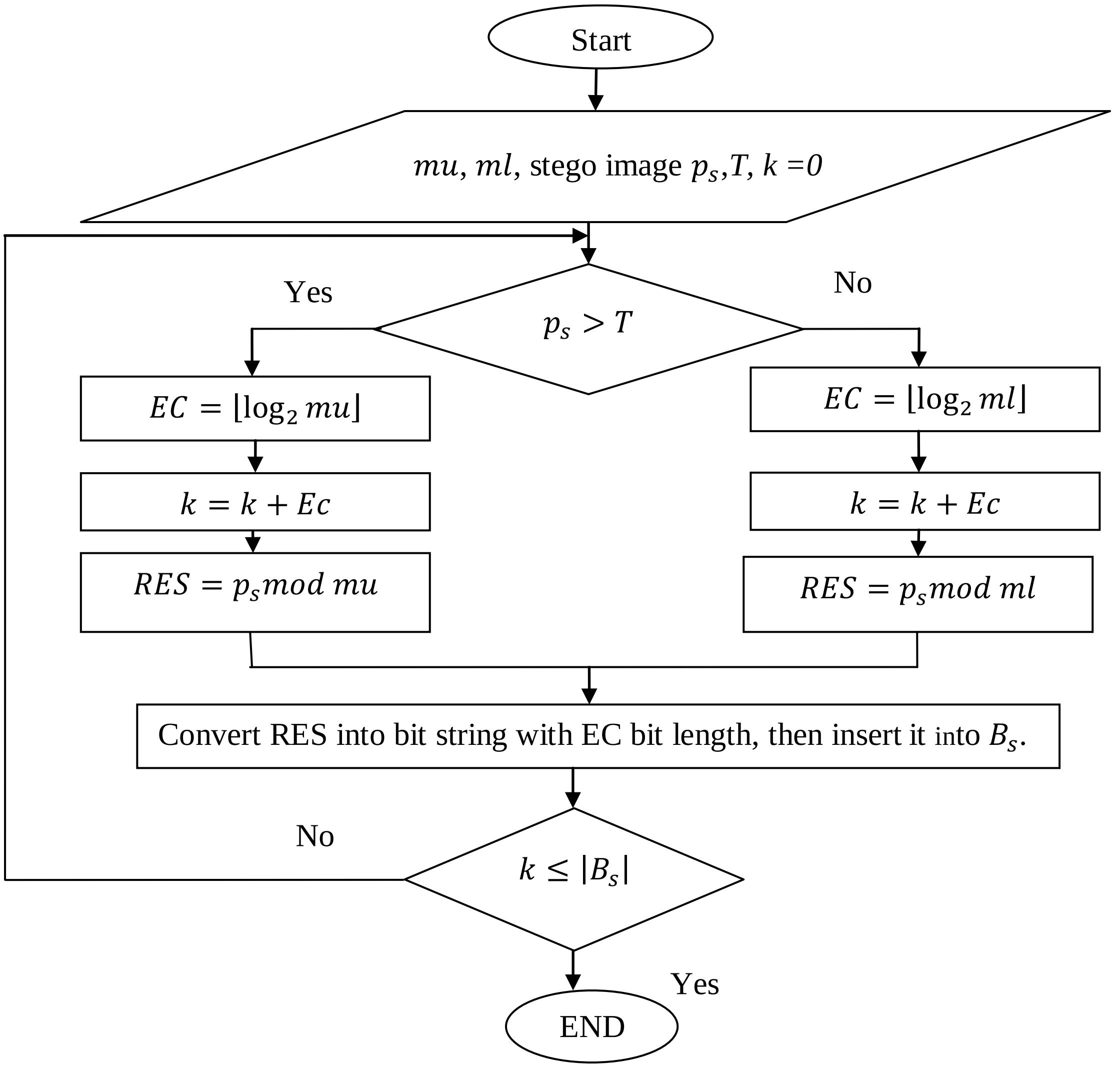


Figure 2.14: Flowchart for LSBT The Extraction Algorithm

• Example 3. Example of LSBT embedding-extraction

Let the cover pixel values be $(160\ 200\ 255\ 150\ )_{10}$ for cover pixel, and B_s be a secret bit string $(01110111101)_2$, T=160, mu=8, ml=4.

Embedding into $p_1 = 160$:

Step 1:

Check if $(p_1 \geq T) = (160 \geq 160)?\, yes$

Hence, calculate EC according to (2.15)

$EC = \lfloor \log_2 8 \rfloor = 3$.then read 3 bits from $B_s, (011)_2$,

and convert them to decimal, $DEC = 3$

Calculate RES according to (2.16)

$RES = 160\ mod 8 = 0.$

Step 2: Compute *D* according to (2.19)

$D = |0 - 3| = 3.$

Step 3: Pixel, $p_1 = 160$, meets case 2 condition since $p_1 = 160\ \geq \mathrm{T} = 160$

And it meets case 2.3 condition since

$T = 160 \leq pij = 160 \leq \mathrm{T} + \frac{\mathrm{mu}}{2} = 164.$

Hence, DEC=3 is embedded according to (2.53):

$p_s(1) = 160 - 0 + 3 = 163.$

Now, embed into $p_2 = 200.$

Step 1:Since $p_2 = 200 \geq T = 160$

calculate EC according to (2.15)

$EC = \lfloor \log_2 8 \rfloor = 3.$

$$EC = 3, then\ read\ 3\ bits\ from\ B_s, (101)_2, and\ convert\ them\ to\ decimal, DEC = 5$$

Calculate RES according to (2.16):

$RES = 200 mod 8 = 0.$

Step 2: Compute *D* according to (2.19)

$D = |0 - 5| = 5.$

Step 3: Embed DEC=5 into $p_2 = 200$

according to case 2.2 since

$T + \frac{mu}{2} = \left(160 + \frac{8}{2}\right) = 164 < pij = 200 < 255 - \frac{mu}{2} + 1 = \left(255 - \frac{8}{2} + 1\right) =$

$252.$

Case 2.2.1 condition holds:

$\mathrm{D} = 5 > \frac{\mathrm{mu}}{2} = 4.$

Hence, calculate AV as (2.41):

$AV = \mathrm{mu} - \mathrm{D} = 8 - 5 = 3.$

Since case 2.2.1.2 holds, according to (2.45)

$p_s(2) = \mathrm{pij} - \mathrm{AV} = 200 - 3 = 197.$

Now, consider embedding into $p_3 = 255$:

Step 1: Since $p_3 = 255 \geq T = 160$

calculate EC according to (2.15)

$EC = \lfloor \log_2 8 \rfloor = 3.$

$$EC = 3, read\ 3\ bits\ from\ B_s(111)_2\ convert\ to\ decimal\ DEC = 7$$

Calculate RES according to (2.16)

$RES = 255 mod 8 = 7.$

Step 2: Compute D according to (2.19)

$D = |7 - 7| = 0.$

Step 3: Embed DEC=7 into $p_3 = 255$

according to case 2.1 since according to (2.38)

$255 > (255 - 8 + 1),$

$p_s(3) = (255 - 8 + 1) + 7 = 255.$

Embedding $(01)_2$ into $p_4 = 150$:

Step 1:Since p4=150<T=160

calculate EC according to (2.17)

$EC = \lfloor \log_2 4 \rfloor = 2,$

$$EC = 2, read\ 2\ bits\ from\ B_s(01)_2\ convert\ to\ decimal\ DEC = 1$$

Calculate RES according to (2.18)

$RES = 150 mod 4 = 2.$

Step 2: Compute D according to (2.19)

$D = |2 - 1| = 1.$

Step 3: Embed DEC=1 into $p_3 = 150$

according to case 1.2 since

$\frac{4}{2} < 150 < 160 - \frac{4}{2},$

$2 < 150 < 158.$

Case 1.2.2 condition holds:

$\mathrm{D} = 1 \leq \frac{\mathrm{mu}}{2} = 2.$

Hence, calculate AV as (2.30):

$AV = \mathrm{D} = 1.$

Since Case 1.2.2.1 holds, according to (2.32)

$$p_s(4) = 150 - 1 = 149.$$

The stego pixel are $(163\ 197\ 255\ 149)_{10}$.

In the extraction, we have stego pixel value $(163\ 197\ 255\ 149)_{10}$, and stego pixel,

$p_s(1)$=163.

According to **Case 2** $p_s(1) \geq T$

From (2.56) and (2.57)

$EC = \lfloor \log_2 8 \rfloor = 3,$

$$RES = 163 mod 8 = 3.$$

Convert RES to binary with EC length: $(011)_2$.

The secret message $(011)_2$ is restored.

The second stego pixel, $p_s(2)$=197.

According to **Case 2** $p_s(2) \geq T$

From (2.56) and (2.57)

$EC = \lfloor \log_2 8 \rfloor = 3.$

$$RES = 197 mod 8=5.$$

Convert RES to binary with EC length $(101)_2$

The secret message $(101)_2$ is restored.

The third stego pixel, $p_s(3)$=255.

According to **Case 2** $p_s(i) \geq T$.

From (2.56) and (2.57)

$EC = \lfloor \log_2 8 \rfloor = 3,$

$$RES = 255 mod 8=7.$$

Convert RES to binary with EC length $(111)_2$

The secret message $(111)_2$is restored.

And the fourth stego pixel, $p_s(4)$=149.

According to **Case 1,** $p_s(i) < T$.

From (2.54) and (2.55)

$EC = \lfloor \log_2 4 \rfloor = 2,$

$$RES = 149 mod 8=1.$$

Convert RES to binary with EC length $(01)_2$

The secret message $(01)_2$ is restored.

Finally, we embedded $(01110111101)_2$ into $(160\ 200\ 255\ 150)_{10}$, and the secret message $(01110111101)_2$ is restored from the stego pixels $(163\ 197\ 255\ 149)_{10}$.

2.4 Color Perception

The human eye distinguishes color by hue, saturation, and brightness. As we know, primary colors can be one-to-one correlated with light wavelength. Also, combinations of different light wavelengths can cause the same perception of color [23].

The RGB color model is a convenient means for representing color which was established in Commission Internationale de l`Eclairage (CIE) in 1931. R, G and B are three light waves with three different colors called reference color stimuli and denote wavelengths of monochromatic light of 700.0 nm for R which is selected from a wavelength region where human perception does not change much with changing wavelength 546.1 nm for G, and 435.8 nm for B that correspond to emission lines of Hg lamp. To establish a broad array of colors by an additive color model, Red, Green, and Blue lights are mixed together in various ways. The color matching experiment mixes three primary colors to find a coincidence with the given color by human perception [20]. Figure 2.15 shows the color matching experiment for RGB.

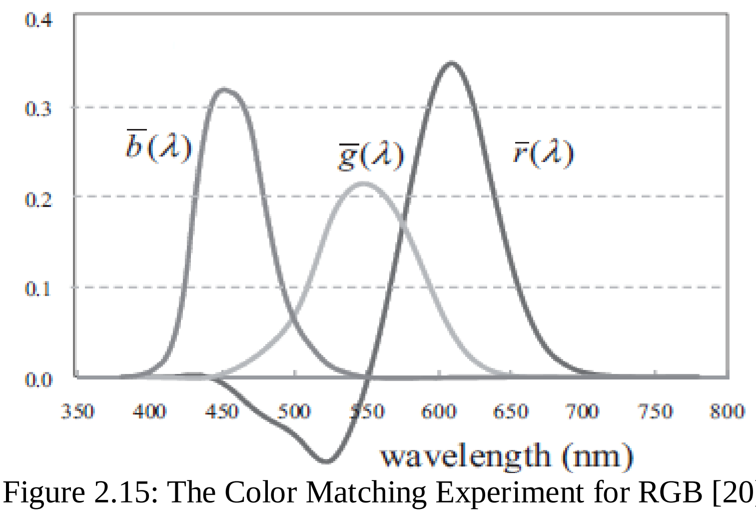


Figure 2.15: The Color Matching Experiment for RGB [20].

2.5 Known Metrics for Evaluating Quality of Stego Images

Several metrics are used usually to measure the quality of the stego-images. In the following, we explain them.

- Mean Square Error (MSE)

MSE is the mean of squares of differences between the cover image and the stego image [28]:

$$Mse = \frac{\sum_{k=1}^{3}\sum_{i=1}^{M}\sum_{j=1}^{N}(X_{ijk}^{c} - X_{ijk}^{s})^2}{3 * m * n} \tag{2.58}$$

where *N, M* is number of columns and rows, respectively, of cover image, X^c, and the stego image, X^s

- Peak Signal to Noise Ratio (PSNR)

PSNR is calculated as follows:

$$PSNR = 10\,log_{10}\frac{255^2}{MSE}\text{ dB}, \tag{2.59}$$

where 255 is the maximum possible value of gray scale pixel value [28]. PSNR uses potentially maximal pixel value, 255, that actually may not be reached, that may result not in proper quality description by it.

- Signal to Noise Ratio (SNR)

Signal to noise ratio refers to the measurement of the level of an audio signal as compared to the level of noise that is present in that signal. A larger value of SNR implies a better quality [3], it is calculated as follows:

$$SNR = 10 * log_{10}\frac{\sigma_{Ri}^2+\sigma_{Gi}^2+\sigma_{Bi}^2}{\sigma_{Rn}^2+\sigma_{Gn}^2+\sigma_{Bn}^2}\text{ dB} \tag{2.60}$$

where σ_{Ri}^2, σ_{Gi}^2, and σ_{Bi}^2 are color component variances of cover image, and σ_{Rn}^2, σ_{Gn}^2, and σ_{Bn}^2 are color component variances of noise-added image[5]. Variance is calculated as follows:

$$\sigma^2 = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(\overline{x}-u_{ij})^2}{MN}, \tag{2.61}$$

where $\overline{x}$ is mean of the pixel image, u_{ij} is the pixel value of an image, and, NM is the number of pixels of the image, and mean is defined as follows:

$$\overline{x} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} x_{ij}}{MN}, \tag{2.62}$$

where $\overline{x}$ is mean of the pixel image, x_{ij} is the pixel value of an image, and, NM is the number of pixels of the image.

- Embedding capacity

Embedding capacity is defined as the number of bits of secret data embedded in each pixel of cover image.

$$EC = \frac{Size\ of\ secret\ data\ in\ bits}{Size\ of\ cover\ image\ in\ bytes}\ BPP. \tag{2.63}$$

2.6 Known Experiments Setups and Results

1) For gray scale images, the eight cover images used in [28], are shown in Figure 2.16, and PSNR for the both schemes(ATD and LSBT), for random secret messages with different size and averaged over 8 cover images Figure 2.16 are shown in Table 2.1, Figure 2.17 shows PSNR for two schemes (ATD and LSBT) [28].

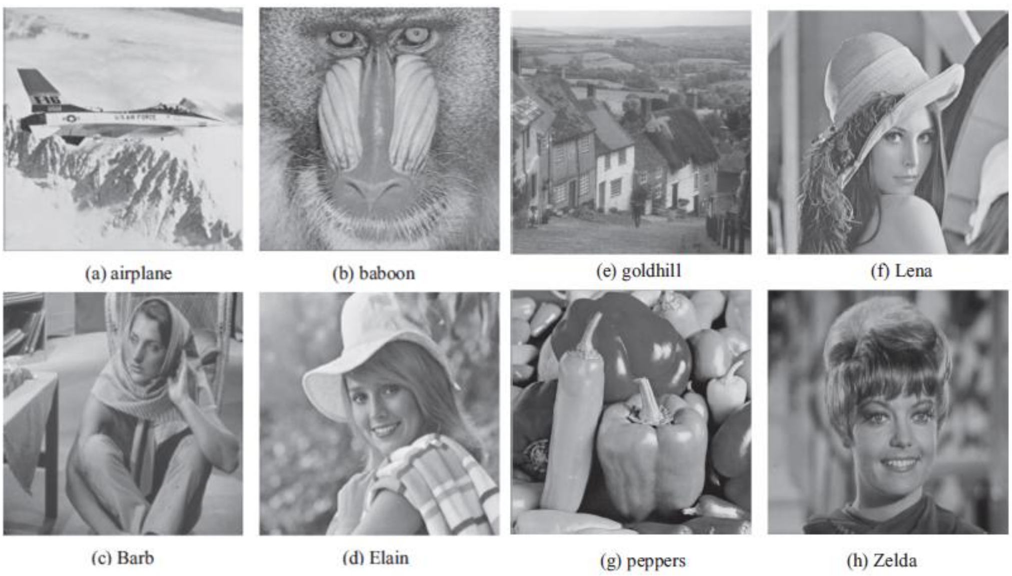


Figure 2.16: Cover Images Used in [28]

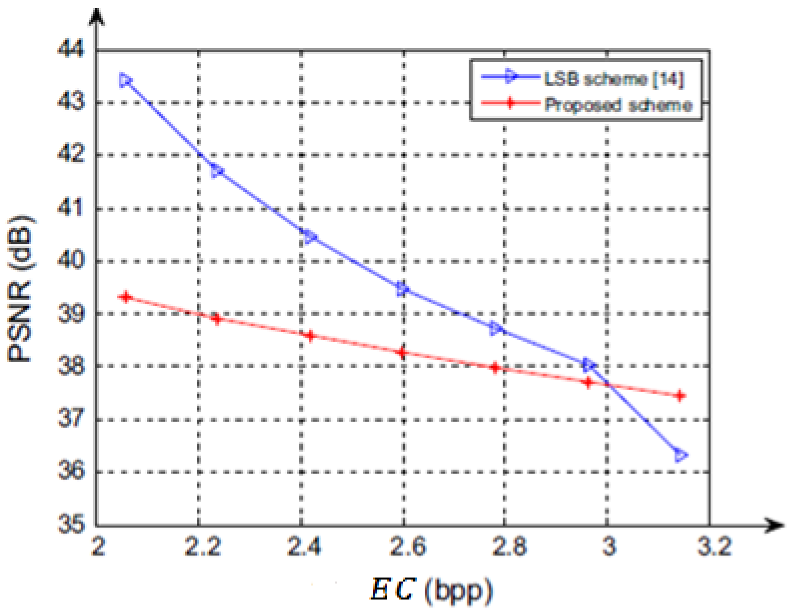


Figure 2.17: PSNR Values (dB) for ATD and LSBT Obtained in [28] for Random Secret Messages with Different Sizes and Averaged Over 8 Cover Images Figure 2.16.

Table 2.1: PSNR (dB) of LSBT and ATD [28] for Random Secret Message with Embedding Capacity 3.1699 BPP for 8 Cover Images Figure 2.16

Cover image	LSBT	ATD
Airplane	36.11	37.45
Baboon	36.10	37.41
Barb	36.09	37.41
Elain	36.11	37.42
Goldhill	36.21	37.41
Lena	36.04	37.41
Peppers	36.07	37.39
Zelda	36.11	37.40

From Table 2.1 and Figure 2.17, we see that the ATD has better image quality when the embedding capacity greater is than 3 BPP, while LSBT achieves higher stego image quality when embedding capacity is less than 3 BPP.

2) For color scale images, Table 2.2 shows PSNR and MSE for LSB algorithm [24], and two cover images used in [24] to embed the secret message are shown in Figure 2.18.

Table 2.2: LSB PSNR (dB) and MSE for LSB [24]

Cover image	PSNR(dB)	MSE
Lena	55.9461	0.1654
Baboon	55.9238	0.1662

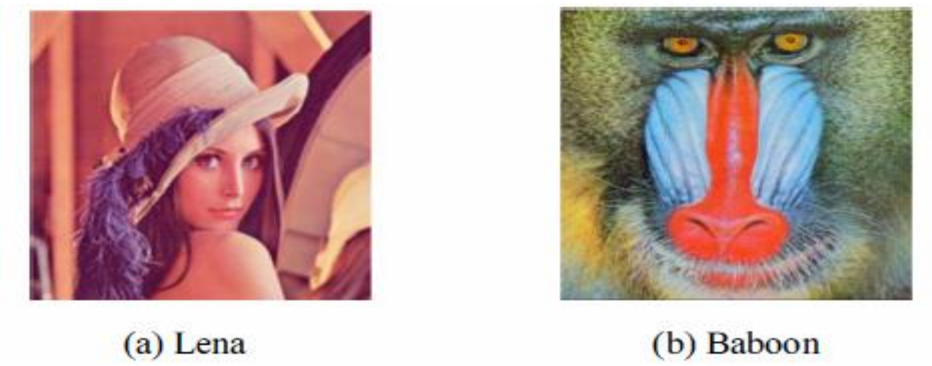


Figure 2.18: Cover Images Used in [24].

2.7 Problem Definition

In this thesis, we study ATD and LSBT for gray scale images, and LSB and ATD for color scale images.

• For gray scale images:

1) For LSBT and ATD, the papers [26] [28] do not provide the proofs of the methods, hence, we prove correctness of ATD

2) For LSBT, information is not provided in [26] how to set the threshold T and two moduli mu, ml. We give examples showing that LSBT [26] works incorrectly in some cases, explain them, and fix the problems by proposing modified LSBT, LSBT-M, for which we define parameter values such that the algorithm works correctly.

3) In [28], PSNR of ATD and LSBT is presented without theoretical explanation,

hence, we find a formula for threshold T dependence of embedding capacity Bit Per Pixel (BPP).

4) In [28], ATD considers the secret message as the ternary stream, but conventional messages are binary, hence, we need solving the problem of binary-ternary conversions.

5) In [28], only 8 cover images were considered in the experiments, hence, we extend experiments to 15 cover images for the both algorithms (ATD, LSBT).

• For color images:

1) In [28], ATD works on the gray scale images only, hence, we extend ATD to work on the color scale images. and we test it for digit embedding combinations (112, 121, 112), where, for example, combination 112 means that, one ternary digit is embedded into R component, one ternary digit is embedded into G component, and two ternary digits are embedded into B component of RGB pixel.

2) We implement LSB for bit embedding combinations (224, 242, 422, 134, 143, 314, 341, 413, 431), where, for example, combination 143, means that, one bit is embedded into R component, four bits are embedded into G component, and three bits are embedded into B component of RGB pixel.

3) We modify LSB to the adaptive LSB, ALSB, which determines bit embedding combination depending on the color intensity of each component R, G, and B.

4) Study PSNR characteristics to find the most appropriate for color scale images quality description. From our experiments for the color scale images, we propose MSNR (Mean Signal to Noise Ratio), APSNR (Actual PSNR) for color scale image components (each of them is a gray scale image).

In MSNR as in (2.64) we calculate the mean value overall pixel image and change the possible peak signal which is 255 in equation (2.59) to the mean value of pixels.

$$MSNR = 10\, log_{10} \frac{\bar{x}}{MSE} \text{dB}, \quad (2.64)$$

Where $\bar{x}$ is the mean value of the pixels of the cover calculated according to (2.62). In APSNR as in (2.64), APSNR considers actual peak signal instead of possible peak signal which is 255 in equation (2.59).

$$APSNR = 10\, log_{10} \frac{X}{MSE} \text{ dB}, \quad (2.65)$$

where X is the actual peak value of the pixels of the cover image.

5) Also, we propose weighted PSNR, APSNR, and MSNR for color scale images by three different groups of weights $(R_w = 0.4,\ G_w = 0.3,\ B_w = 0.3)$,$\left(R_w = \frac{1}{3},\ G_w = \frac{1}{3},\ B_w = \frac{1}{3}\right)$, and $(R_w = 0.4,\ G_w = 0.243,\ B_w = 0.357)$, R_w weights of Red, G_w weights of Green, B_w weights of Blue.

2.8 Summary of Chapter 2

Thus, in this chapter, we have done the following:

1. We have presented the overview of Steganography and methods of Steganography.

2. We have presented the related work and known experiments on LSB, LSBT, and ATD. Also, we explain three algorithms (LSB, ATD, and LSBT) with numerical examples for each of them.

3. We have presented the color perception, and the known metrics for evaluation performance of Steganographic algorithms (LSB, LSBT, and ATD) and the results of known experiments from [28], [24].

4. Problem definition for the thesis is given.

Chapter 3

ATD AND LSBT ANALYSIS AND RECOVERED PROBLEMS FIXING

In this chapter, analysis of ATD and LSBT is given.

3.1 Proof of ATD Correctness

We embed digits by pairs, and the first digit in a pair, even-numbered secret digit is embedded into $sub1\grave{}_{ij}$ by (2.6)-(2.8). After that, $v\grave{}_{ij}$ is defined by (2.9), and the second digit in the pair, odd-numbered digit, is embedded into $v\grave{}_{ij}$ by (2.10)-(2.12). Consider extraction of the first, even numbered secret digit $S\grave{}_k$. It is extracted from $sub1_{ij}$ by (2.13). Consider the following cases:

Case 1: If the secret digit S_k was embedded according to (2.6), then $mod\left(sub1\grave{}_{ij}, 3\right) = S_k$ holds, and due to Note 1,

$$S\grave{}_k = mod\left(sub1_{ij}^{stego}, 3\right) = mod(sub1\grave{}_{ij}, 3) = S_k \qquad (3.1)$$

Hence, extracted digit,$S\grave{}_k$, and embedded digit S_k are the same, $S\grave{}_k = S_k$.

Case 2: If the secret digit, S_k, was embedded according to (2.7), then $mod\left(sub1\grave{}_{ij} + 1,3\right) = S_k$ holds, and due to Note 1,

$$S\grave{}_k = mod\left(sub1_{ij}^{stego}, 3\right) = mod\ (sub1'_{ij} + 1,3) = S_k \qquad (3.2)$$

Hence, extracted digit $S\grave{}_k$, and embedded digit S_k are the same, $S\grave{}_k = S_k$.

Case 3: If the secret digit S_k was embedded according to (2.8), then $mod\left(sub1\grave{}_{ij} - 1,3\right) = S_k$ holds. and due to Note 1,

$$S_k^{`} = mod\ (sub1_{ij}^{stego}, 3) = mod\left(sub1_{ij}^{`} - 1{,}3\right) = S_k \tag{3.3}$$

Hence, extracted digit, $S_k^{`}$, and embedded digit, $S_k$, are the same, $S_k^{`} = S_k$.

As far as embedding of the first digit is done only by three ways, (2.6)-(2.8), and in each one extraction from $sub1_{ij}^{stego}$ is done correctly, i.e., extracted digit is the same as the embedded one, hence, ATD correctness for the first digit in a pair is proved.

Now consider extraction of $S_k^{`}$, an odd numbered secret digit (the second digit in a pair), that is extraction from the pixel value v_{ij}^{stego}.

Embedding into the pixel value the second ternary digit is made by (2.10), (2.11), and (2.12). Consider each of the cases as follows:

Case 1: If the secret digit S_k was embedded according to (2.10), then $mod(v_{ij}^{`}$,3)=S_k and

$$S_k^{`} = mod\left(v_{ij}^{stego}, 3\right) = mod\left(v_{ij}^{`}, 3\right) = S_k \tag{3.4}$$

Hence, $S_k^{`} = S_k$.

Case 2: If the secret digit S_k was embedded according to (2.11), then $mod(v_{ij}^{`}$+1,3)=S_k and

$$S_k^{`} = mod\ (v_{ij}^{stego} + 1{,}3) = mod(vij' + 1{,}3) = S_k \qquad 3.5)$$

Hence, $S_k^{`} = S_k$.

Case 3: If the secret digit S_k was embedded according to (2.12), then $mod(v_{ij}^{`}$-1,3)=S_k and

$$S_k^{`} = mod\left(v_{ij}^{stego} - 1{,}3\right) = mod\left(v_{ij}^{`} - 1{,}3\right) = S_k \tag{3.6}$$

Hence, $S_k^{`} = S_k$.

Thus, as far as in all three cases of embedding, extraction returns the same second digit as embedded, the algorithm correct work is fully proved.

3.2 LSBT Problems Fixing and Proof of Fixed LSBT Correctness

The embedding capacity of a pixel is determined according to the threshold value, T, by (2.15), (2.17). For this algorithm, we prove that, the value of stego pixel still is in the same range as it was in the cover image before embedding, and the algorithm works correctly extracted value is the same as it was embedded.

When the pixel value is less than T, we have three cases for embedding the secret message into a pixel value. We consider each of the cases as follows:

Case 1.1 $p_{ij} < \frac{ml}{2}$

From (2.18)

$$0 \le RES < ml \tag{3.7}$$

From (2.17)

$$0 \le DEC < 2^{EC} \le ml \tag{3.8}$$

From (2.20), (2.21) and (3.8)

$$P_{sij} = DEC < \mathrm{T} \tag{3.9}$$

Thus, P_{sij} and p_{ij} are both in the same, less than *T*. In extraction, from (2.55) and (3.9)

$$RES\text{-}E = DEC \bmod ml$$

Since by (3.8), $DEC < ml$, $RES\text{-}E = DEC$, i.e. extracted and embedded values are the same.

Case 1.2 $\frac{ml}{2} \le p_{ij} < T - \frac{ml}{2}$

Case 1.2.1.1 $\left(D > \frac{ml}{2}\right) and\ (RES \ge DEC)$

From (2.19) and (2.23),

$$D = |RES - DEC| > \frac{ml}{2}. \quad (3.10)$$

From (2.25) and (3.10),

$$DEC < RES - \frac{ml}{2}. \quad (3.11)$$

From (2.18),

$$DEC < p_{ij}\, mod\, ml - \frac{ml}{2} \quad (3.12)$$

From (2.24), (2.25), and (3.10)

$$AV = ml - RES + DEC. \quad (3.13)$$

From (2.26) and (3.13),

$$P_{sij} = p_{ij} + ml - RES + DEC. \quad (3.14)$$

From (2.18), (3.12), and (3.14),

$$P_{sij} = p_{ij} + ml - RES + DEC < p_{ij} + ml - p_{ij}\, mod\, ml + p_{ij}\, mod\, ml - \frac{ml}{2} = p_{ij} + \frac{ml}{2} \quad (3.15)$$

Then, from (3.15) and (2.22),

$$P_{sij} < p_{ij} + \frac{ml}{2} < T. \quad (3.16)$$

Thus, we see that P_{sij} and p_{ij} are both less than *T*.

In extraction algorithm, from (2.18), (2.55), (2.17), and (3.14),

$$RES\text{-}E = (p_{ij} + ml - RES + DEC)mod\, ml = (p_{ij} + ml - p_{ij}\, mod\, ml + DEC)mod\, ml = (p_{ij} - p_{ij}\, mod\, ml)\, mod\, ml + ml\, mod\, ml + DEC\, mod\, ml = 0 + 0 + DEC\, mod\, ml = DEC$$

Thus, extracted value, *RES-E*, and embedded value, *DEC*, are the same.

Case 1.2.1.2 $RES < DEC$

From (2.19) and (2.23),

$$D = |RES - DEC| > \frac{ml}{2} \quad (3.17)$$

From (2.27) and (3.17),

$$DEC > RES + \frac{ml}{2} \tag{3.18}$$

From (2.18) and (3.18),

$$DEC > p_{ij}\, mod\, ml + \frac{ml}{2} \tag{3.19}$$

From (2.24), (2.27), and (3.17)

$$AV = ml - \text{DEC} + \text{RES}. \tag{3.20}$$

From (2.28) and (3.20),

$$P_{sij} = p_{ij} - ml - \text{RES} + \text{DEC} \tag{3.21}$$

From (2.17), (2.18), (2.22), and (3.21),

$$P_{sij} = p_{ij} - ml - p_{ij}\, mod\, ml + DEC < pij - ml + \text{DEC} < pij < T \tag{3.22}$$

Thus, from (3.22),

$$P_{sij} \leq T \tag{3.23}$$

Thus, from (3.23), we have that both P_{sij} and p_{ij} are less than T.

In extraction algorithm, from (3.22), (3.23),(2.17), and (2.55),

$$RES\text{-}E = (p_{ij} - p_{ij}\ mod\ \ ml)\ mod\ \ ml - ml\ mod\, ml + DEC\ mod\ ml$$

$$= 0 - 0 + DEC\ mod\ ml \quad = DEC$$

Thus, extracted *RES-E* value and embedded value, *DEC*, are the same.

Case 1.2.2.1

From (2.19) and (2.29)

$$D = |RES - DEC| \leq \frac{ml}{2} \tag{3.24}$$

From (2.31) and (3.24),

$$DEC \geq RES - \frac{ml}{2} \tag{3.25}$$

From (2.18),

$$DEC \geq p_{ij}\, mod\, ml - \frac{ml}{2} \tag{3.26}$$

From (2.30), (2.31), and (3.24),

$$AV = RES - DEC \tag{3.27}$$

From (2.32) and (3.27),

$$P_{sij} = p_{ij} - RES + DEC \tag{3.28}$$

From (2.18), (2.30), (2.22), (2.31), and (2.32),

$$P_{sij} = p_{ij} - p_{ij}\, mod\, ml + DEC \leq p_{ij} < T \tag{3.29}$$

Thus,

$$P_{sij} \leq T. \tag{3.30}$$

Hence, from (2.22) and (3.30), both p_{ij} and P_{sij} are less than T.

In extraction algorithm, from (3.28), (3.29), (2.17), and (2.55)

$$RES - E = (p_{ij} - p_{ij}\ \ mod\, ml)\ \ mod\, ml +\ DEC\, mod\, ml$$

$$=0+DEC\ mod\ ml \quad = DEC$$

Hence, extracted value, *RES*-*E*, and embedded value, *DEC*, are the same.

Case 1.2.2.2

From (2.19) and (2.29)

$$D = |DEC - RES| \leq \frac{ml}{2} \tag{3.31}$$

From (2.33) and (3.31),

$$DEC \leq RES + \frac{ml}{2} \tag{3.32}$$

From (2.18) and (3.32),

$$DEC \leq p_{ij}\, mod\, ml + \frac{ml}{2} \tag{3.33}$$

From (2.30), (2.33), and (3.31),

$$AV = DEC - RES \tag{3.34}$$

From (2.34) and (3.34),

$$P_{sij} = p_{ij} + DEC - RES \tag{3.35}$$

From (2.18), (2.22), (3.33), and (3.35),

$$P_{sij} = p_{ij} - p_{ij}\ mod\ ml + DEC < pij - p_{ij}\ mod\ ml + p_{ij}\ mod\ ml + \frac{ml}{2} = \quad (3.36)$$

$$p_{ij} + \frac{ml}{2} < T$$

Then, from (3.36),

$$P_{sij} < T. \quad (3.37)$$

Hence, from (3.37) and (2.22), both $pij\ and P_{sij}$ are less than *T*.

In extraction algorithm, from (2.55), (2.17), and (3.35),

$$RES\text{-}E = (p_{ij} - p_{ij}\ mod\ ml)\ mod\ ml + DEC\ mod\ ml \quad =$$

$$= 0 + DEC\ mod\ \ ml \quad = DEC.$$

Thus, extracted value, *RES-E*, and embedded value, *DEC*, are the same.

Case 1.3 Condition (2.35) holds.

From (2.17),

$$0 \leq DEC < 2^{EC} \leq ml \quad (3.38)$$

From (2.18), (2.35), and (2.36),

$$P_{sij} = p_{ij} - p_{ij}\ mod\ ml + DEC, \quad (3.39)$$

$$p_{ij} < T, \quad (3.40)$$

$$p_{ij} - p_{ij}\ mod\ ml = k\ ml. \quad (3.41)$$

Using (3.41) in (3.39), we get:

$$P_{sij} = k\ ml + DEC. \quad (3.42)$$

Let us construct a counterexample, showing that P_{sij} (3.42) may be not less than *T* contrary to p_{ij} meeting (2.35).

Counterexample 1. Let the threshold value *T*=160, cover pixel p_{ij}=159, ml =9, and a secret message is $(111)_2$.

From (2.18),

$$EC = \lfloor \log_2 9 \rfloor = 3.$$

From (2.19),

$$RES = 159\ mod\ 9 = 6.$$

DEC is the decimal value of *EC=3* bit length, and the secret message is $(111)_2$; then, *DEC*=$(7)_{10}$.

From (2.35) and (2.36),

$$P_{sij} = 159 - 6 + 7 = 160 = T.$$

Thus, Counter example 1 shows that the value of stego pixel, P_{sij}, is 160 that is not less than *T*, whereas original cover pixel value, $p_{ij} = 159$, is less than *T*.

According to the Counter example 1, we suggest the first amendment of LSBT method

LSBT Amendment 1. Let T and ml in LSBT input satisfy the following condition (3.43):

$$T = k1 * ml, \tag{3.43}$$

where *k1* is an integer.

Proof. Let us prove that in the condition of Counter example 1, when (3.42) holds, LSBT with Amendment 1 works correctly. From (3.40), (3.41),

$$k < \text{k1}. \tag{3.44}$$

Hence, from (3.43), (3.44) and (3.42),

$$P_{sij} = kml + DEC < kml + ml = (k+1)\ ml \le T = k1 * ml. \tag{3.45}$$

In extraction algorithm, from (3.45), (2.17), and (2.55),

$$RES - E = P_{sij}\ mod\ ml = \tag{3.46}$$

$$=(kml + DEC)\ mod\ ml = kml\ mod\ ml + DEC\ mod\ ml = DEC \tag{3.47}$$

Thus, extracted value, *RES-E,* and embedded value, *DEC,* are the same, and LSBT with Amendment 1 works correctly in the conditions of the counter example 1 resulting from (3.42). All the proofs, preceding Counter example 1, are valid for LSBT with Amendment 1 since no condition was imposed in LSBT on T and ml.

Now, continue the proof of LSBT with Amendment 1 correctness.

When the pixel value is larger than or equal to T, we have the three cases for embedding the secret message into the pixels. We consider the cases below:

Case 2.1 $p_{ij} \geq$ T, $p_{ij} > 255 - \frac{mu}{2} + 1$.

From (2.16),

$$0 \leq RES = p_{ij} mod\, mu < mu. \tag{3.48}$$

From (2.15),

$$0 \leq DEC < 2^{EC} \leq mu. \tag{3.49}$$

From (2.37) and (2.38),

$$P_{sij} = 256 - mu + DEC. \tag{3.50}$$

We face difficulty in proving that P_{sij} (3.50) is not less than *T* as pij for the Case 2. We show in Counter example 2 that P_{sij} meeting (3.50), may, contrary to p_{ij}, be less than the threshold, *T*.

Counter example 2. Let $T = 252$, $mu = 18$, $p_{ij} = 253 > 256 - \frac{mu}{2} = 256 - 9 = 247$, $DEC = 1$, then $P_{sij} = 256 - mu + DEC = 256 - 18 + 1 = 237 < T = 252$.

Hence, P_{sij} is less than *T*, where as original p_{ij}>*T*.

In Counter example 2, *T* is an integer multiple of mu, and mu is not a power of 2. Let us consider one more counter example with *T* not being a multiple of mu, and mu being a power of 2.

Counter example 3. Let $T = 252$, $mu = 16$, $p_{ij} = 253 > 256 - mu/2 = 256 - 8 = 248$, $DEC = 1$, then P_{sij}=256- mu + DEC $= 249 < T = 252$.

Hence, P_{sij} is less than *T*, where as p_{ij}>*T*.

To fix the problem seen from Counter examples 2, 3, let us consider the second amendment of LSBT.

LSBT Amendment 2. Let T be a multiple of mu, that is a power of 2:

$$\mathrm{T} = k1 * mu, \tag{3.51}$$

$$mu = 2^{k2}, \tag{3.52}$$

where $k1$, $k2$ are some positive integers.

Proof of the LSBT Amendment 2 correctness. Let us prove that when (3.50) in the conditions of the Case 2.1, LSBT works correctly if (3.51), (3.52) hold. From (2.37), and assuming that 256= $k3$* mu,

$$p_{ij} > 256 - \frac{mu}{2} = k3 * mu - \frac{mu}{2} = (k3 - \frac{1}{2}) * mu, \tag{3.53}$$

$$p_{ij} \geq T = k1 * mu. \tag{3.54}$$

Since $T < 256$,

$$k1 < k3. \tag{3.55}$$

According to (3.50), (3.54), (3.55)

$$P_{sij} = 256 - mu + DEC = (k3 - 1) * mu + DEC \geq k1 * mu + DEC \tag{3.56}$$
$$= T + DEC \geq T$$

From (3.56), we see that both, P_{sij} and p_{ij}, are not less than *T*.

In extraction algorithm, from (2.56), (2.57), (3.50), and (3.51)

$$RES - E = (256 - mu + DEC)\ mod\ mu \quad = \tag{3.57}$$
$$\big((k3 - 1)mu + DEC\big) mod\ mu \ = \ DEC\ mod\ mu \ = DEC.$$

From (3.57), the extracted value, *RES-E*, and embedded value, *DEC*, are the same. Since LSBT Amendment 2 does not affect conditions of LSBT Amendment 1, we can conclude, that LSBT with Amendments 1 and 2 works correctly in the conditions of Cases 1 and 2.1 of LSBT Embedding Algorithm.

Let us continue proving of LSBT with Amendments 1, 2 for the rest cases of LSBT Embedding Algorithm.

Case 2.2.1.1. $RES > DEC$

From (2.19) and (2.40),

$$D = |RES - DEC| > \frac{mu}{2}. \tag{3.58}$$

From (2.42) and (3.58),

$$DEC < RES - \frac{mu}{2}. \tag{3.59}$$

From (2.16),

$$DEC < p_{ij}\ mod\ mu - \frac{mu}{2}. \tag{3.60}$$

From (2.41), (2.42), and (3.58),

$$AV = mu - RES + DEC. \tag{3.61}$$

From (2.43) and (3.61)

$$P_{sij} = p_{ij} + mu - RES + DEC. \tag{3.62}$$

From (2.16), (2.39), and (3.62),

$$P_{sij} = p_{ij} - p_{ij}\ mod\ mu + mu + \mathrm{DEC} > p_{ij} + DEC \geq p_{ij} > T. \tag{3.63}$$

Then, from (3.63),

$$P_{sij} > T\ . \tag{3.64}$$

Thus, both P_{sij} and p_{ij} are greater than *T*.

In extraction algorithm, from (2.15), (2.39), (3.63), and (2.57),

$$RES\text{-}E = (p_{ij} - p_{ij}\ mod\ mu)\ mod\ mu + mu\ mod\ mu + DEC\ mod\ mu$$

$$0 + 0 + DEC\ mod\ mu \quad = DEC.$$

Thus, the extracted value, *RES-E*, is the same as the embedded value, *DEC*.

Case 2.2.1.2 $RES < DEC$

From (2.19) and (2.40),

$$D = |RES - DEC| > \frac{mu}{2}. \tag{3.65}$$

From (2.44) and (3.60),

$$\mathrm{DEC} > RES + \frac{mu}{2}. \tag{3.66}$$

From (2.16) and (3.66),

$$DEC > p_{ij}\, mod\, mu + \frac{mu}{2}. \quad (3.67)$$

From (2.41), (2.44), and (3.60),

$$AV = mu - DEC + RES. \quad (3.68)$$

From (2.45) and (3.68),

$$P_{sij} = p_{ij} - mu - RES + DEC. \quad (3.69)$$

From (2.16),(3.67), and (3.69),

$$P_{sij} > p_{ij} - p_{ij}\, mod\, mu - mu + p_{ij}\, mod\, mu + \frac{mu}{2}. \quad (3.70)$$

Then, from (3.70) and (2.39),

$$P_{sij} > p_{ij} - \frac{mu}{2} > T. \quad (3.71)$$

From (3.71) and (2.39), both P_{sij}, p_{ij} are greater than *T*.

In the extraction algorithm, from (2.57), (3.69), (2.15), and (2.16),

$$RES\text{-}E = (p_{ij} - p_{ij}\ mod\ mu)\ mod\ mu - mu\ mod\ mu + DEC\ \ mod\ mu\ \ =$$

$$0 + DEC\ mod\ mu\ \ = DEC.$$

Thus, the embedded value, *DEC*, is the same as the extracted value, *RES-E*.

Case 2.2.2.1 $D \leq \frac{mu}{2}, RES \geq DEC$

From (2.19) and (2.46),

$$D = |RES - DEC| \leq \frac{mu}{2}. \quad (3.72)$$

From (2.48) and (3.72),

$$DEC \geq RES - \frac{mu}{2}. \quad (3.73)$$

From (2.16) and (3.73),

$$DEC \geq p_{ij}\, mod\, mu - \frac{mu}{2}. \quad (3.74)$$

From (2.47), (2.48), and (2.19),

$$AV = RES - DEC. \quad (3.75)$$

From (2.49) and (3.75),

$$P_{sij} = p_{ij} - RES + DEC. \tag{3.76}$$

From (2.16), (3.72), and (3.74),

$$P_{sij} \geq p_{ij} - p_{ij}\, mod\, mu + p_{ij}\, mod\, mu - \frac{mu}{2} = p_{ij} - \frac{mu}{2} \tag{3.77}$$

Then according to (2.39), (3.77),

$$P_{sij} \geq p_{ij} - \frac{mu}{2} > T. \tag{3.78}$$

Thus, from (2.39), (3.78), both P_{sij} and p_{ij} are greater than *T*.

In the extraction algorithm, from (3.76), (2.15), (2.16), and (2.57),

$$RES\text{-}E = (p_{ij} - p_{ij}\ mod\ \ mu)\ mod\, mu + DEC\ \ mod\, mu$$

$$= 0 + DEC\ \ mod\, mu\ \ = DEC.$$

Thus, the extracted value, *RES-E*, is the same as the embedded value, *DEC*.

Case 2.2.2.2. $RES < DEC$.

From (2.19) and (2.46),

$$D = |DEC - RES| \leq \frac{mu}{2}. \tag{3.79}$$

From (2.50) and (3.79),

$$DEC \leq RES + \frac{mu}{2}. \tag{3.80}$$

From (2.16),

$$DEC \leq p_{ij}\, mod\, mu + \frac{mu}{2}. \tag{3.81}$$

From (2.47), (3.79),and (2.50),

$$AV = D = DEC - RES \geq 0. \tag{3.82}$$

From (2.51), (2.39), and (3.82),

$$P_{sij} = p_{ij} + \mathrm{D} = p_{ij} + \mathrm{DEC} - \mathrm{RES} \geq p_{ij} > T. \tag{3.83}$$

Thus, from (2.39), (3.83), both P_{sij}, p_{ij} are greater than *T*.

In the extraction algorithm, from (3.82), (2.15) and (2.16),

$$RES\text{-}E = (p_{ij} - p_{ij}\ mod\, mu)\ mod\, mu + DEC\ \ mod\, mu\ \ =$$

$$0 + DEC\ \ mod\ \ mu\ \ = DEC.$$

Thus, the extracted value, *RES-E*, is the same as the embedded value, *DEC*.

Case 2.3 $T \le p_{ij} < T + \frac{mu}{2}$.

From (2.16) and (2.53),

$$P_{sij} = p_{ij} - p_{ij}\, mod\, mu + DEC. \tag{3.84}$$

$$p_{ij} - p_{ij}\, mod\, mu = k * mu \tag{3.85}$$

$$P_{sij} = k * mu + DEC \tag{3.86}$$

Due to LSBT Amendment 2, (3.51) holds. Hence, from (2.16), (3.85), and (2.52),

$$T + \frac{mu}{2} = k1 * mu + \frac{mu}{2} > p_{ij} = k * mu + RES \ge \mathrm{T} = k1 * mu. \tag{3.87}$$

From (3.87),

$$k1 * mu \le k * mu \le k * mu + RES < (k1 + 0.5)mu < (k1 + 1)mu. \tag{3.88}$$

From (3.88) and (2.16),

$$k1 \le k < k1 + 1. \tag{3.89}$$

From (3.89),

$$k = k1. \tag{3.90}$$

From (3.51), (3.86), and (3.90),

$$P_{sij} = k * mu + DEC = k1 * mu + DEC = T + DEC. \tag{3.91}$$

From (2.15) and (3.91),

$$P_{sij} = T + DEC \ge T. \tag{3.92}$$

Thus, from (3.92), we have that both stego-pixel value, P_{sij}, and original cover pixel value, p_{ij}, are not less than T.

In the extraction algorithm, from (2.52), (3.51), (3.92), and (2.15),

$$RES - E = P_{sij}\, mod\, mu = \tag{3.93}$$

$$=(T + DEC)\, mod\, mu = (\mathrm{k1} * \mathrm{mu} + DEC)\, mod\, mu = 0 + DEC\, mod\, mu = DEC.$$

Hence, from (3.93), the extracted value, *RES-E*, is the same as the embedded value, *DEC*. Thus, we have proved that:

1. The LSBT algorithm works incorrectly under settings specified in [28]. It is proved by Counter examples 1-3 is showing that stego pixel value and original pixel value may belong to different ranges (not less than *T*, less than *T*, where *T* is the threshold value) that generally leads to the extracted value different from the embedded value.

2. The LSBT has fixed by the proposed LSBT Amendments 1, 2, imposing conditions (3.43), (3.51), and (3.52). Under these conditions, we prove that LSBT with Amendments 1, 2 works correctly. LSBT modified with the Amendments 1, 2 we call LSBT-M.

3. Settings used for the experiments in [28] satisfy the conditions we have established in the LSBT-M (see conditions (3.43), (3.51), and (3.52)).

3.3 Summary of Chapter 3

In this chapter, we have proved correctness of ATD, and shown that LSBT is incorrect for its original settings as well as we solve LSBT problem by conditions (3.43), (3.51), and (3.52) on *T*, $\boldsymbol{mu}$, and $\boldsymbol{ml}$; LSBT settings used in the experiments [28] satisfy the conditions of LSBT-M.

Chapter 4

IMPLEMENTATION OF ATD AND LSBT-M ALGORITHMS FOR GRAY SCALE IMAGES

Now, we will present implementation of ATD and LSBT-M algorithms.

4.1 ATD Implementation

We generate a message randomly as secret data, the size of secret data started with 524288 =512 * 512 * 2 bits. In every iteration, it increases by 30,000 bits and we record the values of MSE, PSNR, MSNR, and APSNR. The secret message is divided into non-overlapping blocks the size of block 64-bit; blocks that are converted, at first, to the decimal numeral system, and then to base 3. We receive in the result of the conversion a block of 41 ternary digits.

A secret message is generated by the MATLAB function 'randi' as follows:

```
secret_massege= randi([0 1],1,start_length);
```

A secret message is converted to ternary by the following function:

```
ternary_number=convertto__ternary(secret_massege);
```

In Appendix A.2, lines 1-37, in this function, the secret data are divided into blocks with size 64 bits; then each 64-bit block is converted to 41-ternary-digit block. The first function, in line 11, converts a block to decimal system, and the second function, in line 12, converts the result of the first function, in line 11, to ternary. The full code is displayed in Appendix A.2. After preparing the secret data (by converting it to ternary numbers), we embed two ternary digits in each pixel of a cover image.

Then according to Step 2 in ATD as in Appendix A.3, lines 4-13, in line 7 we convert each pixel of the cover image to the base-2 numeral system (8 bits/pixel) according to (2.3). After that, we divide each pixel into two parts, the first sub segment consisting of the first six bits of the 8 bits (six most significant bits), and the second sub segment consisting of the last two bits (two least significant bits). Then, we modify the pixel according to (2.4), (2.5), as Appendix A.3, lines 14-25.

When embedding the secret message, there is a possibility of overflow/ underflow. An overflow may yield values of some pixel greater than 255 after embedding. The underflow may yield the values of some pixel less than 0. We avoid them by adding one to the sub-segment (for underflow), or subtracting one from the sub-segment (for overflow), as shown in (2.4) and (2.5). For example,

$$v_1 = (11111100)_2, \mathrm{sub}_1 = (111111)_2, \mathrm{sub}_2 = (00)_2$$

Assume the secret digits are (1,2), and to embed this message into the pixel v, we should add 1 to $sub_1 = (111111)_2$ and subtract 1 from $sub_2 = (00)$ for embedding the secret data (1,2) as in (2.7), (2.12), but this operation will cause the overflow. Hence, by preprocessing operators (2.4), (2.5), we get $\mathrm{sub}_1^{`} = (111110)_2$ and $\mathrm{sub}_2^{`} = (01)_2$. Then, we embed two ternary digits into the pixel as (2.6) to (2.11), the first digit is embedded according to (2.6)- (2.8) as shown in Appendix A.3, lines 27-33, and then the second digit is embedded according to (2.9)-(2.11) as shown in Appendix A.3, lines 35-44.

After embedding all of the secret data into the cover image, we record the values of PSNR, MSNR, APSNR, and BPP as shown in Appendix A.1, lines 20-31, in lines 20, and 21, we calculate the mean value of the cover image for calculating the

MSNR (2.64) line 29. In line 28, we calculate the maximal value of the cover image for calculating the APSNR. In lines 27 and 31, we calculate the PSNR (2.59) and APSNR (2.65), respectively. The full code is given in Appendix A.1 and the results are available in Appendix A.6.

As a sample output of our implementation, and the results for embedding the secret message in the cover image (Lena) are shown in Figures 4.1 and Table 4.2.

Cover image Stego image

Figure 4.1: Lena Cover Image and Stego Image for ATD Implementation with Secret Message Size 644288 Bits, Appendix A.6

Table 4.1: ATD Results for Lena Image

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
lena.BMP	39.341881	38.994399	33.082488	7.566452	524288	2.000000
lena.BMP	39.093979	38.746497	32.834586	8.010921	554288	2.114441
lena.BMP	38.863923	38.516441	32.604530	8.446720	584288	2.228882
lena.BMP	38.662911	38.315429	32.403518	8.846863	614288	2.343323
lena.BMP	38.435312	38.087830	32.175919	9.322861	644288	2.457764
lena.BMP	38.245145	37.897663	31.985753	9.740154	674288	2.572205
lena.BMP	38.055621	37.708139	31.796228	10.174622	704288	2.686646
lena.BMP	37.874323	37.526841	31.614930	10.608356	734288	2.801086
lena.BMP	37.695110	37.347628	31.435717	11.055271	764288	2.915527
lena.BMP	37.531930	37.184449	31.272538	11.478558	794288	3.029968
lena.BMP	37.396861	37.049379	31.137469	11.841160	824288	3.144409

For extraction stage, we extract a ternary message by extraction function, Appendix

A.4, lines 1-51. In line 7 is the call the MATLAB function dec2bin to convert the pixel of stego image from decimal to binary, line 8-10, we divide each binary byte (8 bits) into two parts: the first sub-segment consists of the first six bits of the 8 bits, and the second sub-segment consists of the last two bits. And then, we extract the first secret ternary digit from the first sub-segment (2.13), line 12 as well as the second secret ternary digit extract from the stego-pixel (2.14), line 14.

After extracting all the ternary message, we divide the ternary message into blocks with size 41 ternary digits, line 29. After that, we convert each sequence of 41 ternary digits to decimal, line 48, then convert the decimal to binary, line 49. From each pixel, we extract two ternary digits, we continue until all the secret messages are extracted and then convert it to the binary. The text of the code can be viewed in Appendix A.

4.2 LSBT-M Implementation

This algorithm LSBT-M is applied with two different types of threshold:

1. Constant threshold (T=160) (LSB with Constant Threshold, LSBCT), as in [28], Appendix B.
2. Dynamic threshold according to the next formulation (4.1) (LSB with Dynamic Threshold, LSBDT), Appendix C:

$$T = \begin{Bmatrix} \lfloor 160 * (\lceil BPP \rceil - x) + 10 \rfloor & when\ \lceil BPP \rceil \neq x \\ T = 160 & Otherwise \end{Bmatrix}, \quad (4.1)$$

where $\lceil BPP \rceil$ is the maximal embedding capacity, $\lceil BPP \rceil = 3.14$, x is the number of secret bits embedded in each pixel of cover image in each loop calculated by using (2.63). The LSBDT works correctly by imposing conditions on LSBT-M regarding

ml and mu (3.43), (3.51), and (3.52); according to these conditions, the value of T is a multiple of ml and mu, while mu is a power of 2. The code to calculate T is in Appendix C.1, lines 17-30, in line 17, Appendix C.1, we calculate the value of BPP according to (2.63), in lines 18-22, calculate the value of T according to (4.1). As well as in line 23, we check that the value of T meets the conditions of LSBDT.

In this algorithm, we use two moduli numbers mu and ml to determine how many bits will be embedded in the pixel of a cover image according to the value of a threshold. If the value of a pixel is larger than or equal to the threshold, we use mu, otherwise ml. From Appendix B.2, line 2-16. we calculate EC, RES, and D according to (2.15) and (2.17) for EC in line 4, (2.16) and (2.18), for RES line 5, (2.19), for D line 16.

In embedding stage, Appendix B.2, lines 11-25, we check the value of pixel of the cover image; if the pixel value is greater than or equal to the threshold, we use mu, otherwise we use the ml. In line 13, compute *EC* detecting how many bits will be embedded in each pixel as in (2.15) and (2.17). And in line 14, we compute the residue *RES* as in (2.16) and (2.18). *DEC* is the decimal value of *EC* bit fetched from the secret message. In line 24, we compute the *DEC* by the MATLAB function bin2dec. Line 25 calculates *D* from *DEC* and *RES* according to (2.19). Then we embed the secret message according to Case 2 in LSBT:

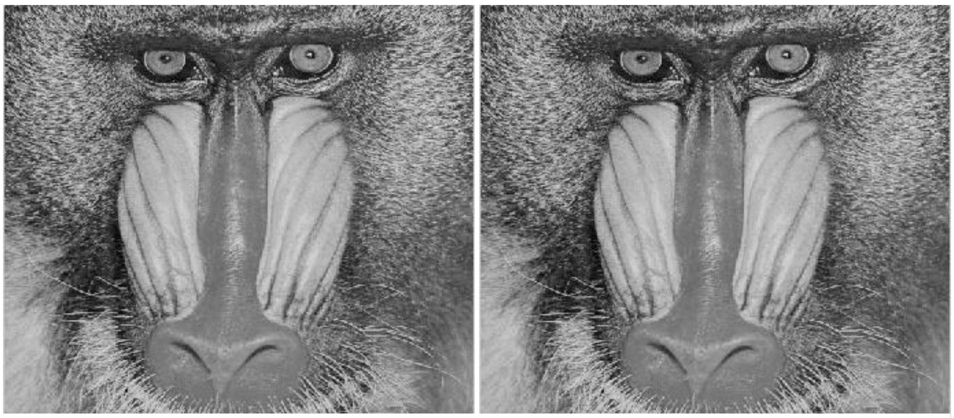
In embedding part, appendix B.2, lines 26-50, we have two cases.

- The pixel of a cover image is greater than or equal to the threshold.
- The pixel of a cover image less than the threshold.

In each case, we have three different situations to compute the value of pixel of stego image. The three different situations for the first case are presented in Appendix B.2, lines 26, 29, and 47 according to (2.37), (2.39), and (2.52), respectively. Also, three different situations for the second case are presented in the Appendix B.2, lines 66, 69, and 87 as in (2.20), (2.22), and (2.35), respectively.

After embedding all of the secret data into the cover image, the values of PSNR (2.59), MSNR, and APSNR are recorded (see Appendix B.1, lines 5-6 and 22-32). In line 6, we calculate the average value of the cover image to be used in calculating MSNR in line 30, and in line 29, we calculate the maximal value of the cover image for calculating APSNR. In lines 28 and 32, we calculate the PSNR and APSNR respectively. The full code is given in Appendix B.1 and the results are available in Appendices B.4 and C.2.

As a sample output of our implementation, the results of embedding the secret message in the cover image (Baboon) are shown in Figure 4.2 and Table 4.2 for LSBDT, and Table 4.3 show results for LSBCT, with T=160.



Cover image   Stego image

Figure 4.2: Baboon Cover Image and Stego Image for LSBDT Implementation

Table 4.2: LSBDT Implementation for Baboon Image

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
Baboon.bmp	524288	43.637205	43.037940	37.728041	2.814232	4	8	192
Baboon.bmp	584288	41.491961	40.892697	35.582797	4.611946	4	8	160
Baboon.bmp	644288	39.981432	39.382167	34.072268	6.530361	4	8	128
Baboon.bmp	704288	38.857527	38.258263	32.948363	8.459167	4	8	96
Baboon.bmp	734288	38.378209	37.778944	32.469045	9.446251	4	8	64
Baboon.bmp	764288	38.039312	37.440047	32.130148	10.212902	4	8	32
Baboon.bmp	824288	35.435728	34.836463	29.526564	18.599770	8	16	160

Table 4.3: LSBCT Implementation for Baboon Image; *T*=160

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
Baboon.bmp	524288	41.955945	41.356681	36.046782	4.144630	4	8	160
Baboon.bmp	584288	41.486587	40.887323	35.577423	4.617657	4	8	160
Baboon.bmp	644288	36.553475	35.954210	30.644311	14.379139	8	16	160
Baboon.bmp	704288	36.170410	35.571146	30.261246	15.705051	8	16	160
Baboon.bmp	734288	35.985098	35.385834	30.075934	16.389683	8	16	160
Baboon.bmp	764288	35.784355	35.185090	29.875191	17.165043	8	16	160
Baboon.bmp	824288	35.431005	34.831741	29.521841	18.620007	8	16	160

In the extraction part, for extracting the secret message from the stego image, we need the value of threshold, *T*, and values of two moduli mu and ml.

As mentioned before, in Appendix B.3, lines 8-52, we compare the value of the pixel with the threshold value, if the threshold is greater than or equal to the pixel value, we use mu, else we use ml. In lines 9 and 29, we compute the value of *EC* by (2.15) and (2.17), and in lines 10 and 30, *RES* is calculated by (2.16) and (2.18). Then, we convert *RES* to decimal with *EC* bits. This is the part of code when the pixel value is greater than the threshold, the all code is in Appendix B.3, and results are in

Appendices B.4 and C.2.

4.3 Summary of Chapter 4

Thus, in this chapter, we have implemented and tested ATD, LSBCT, and LSBDT methods. In the ATD method, we take 'Lena' image as an example, and we take as example 'Baboon' image in LSBDT and LSBCT. The results and all codes of methods for all functions are available in the Appendices A, B, and C.

Chapter 5

IMPLEMENTATION OF ATD AND LSB ALGORITHMS FOR COLOR IMAGES

In this chapter, we present implementation of LSB and ATD for color images. We use 8 color cover images with size 512*512 pixels and one binary message as a secret message. We use MSE (2.58), PSNR (2.59), SNR (2.60), WPSNR, MSNR (2.64), and WMSNR for evaluating the performance of the both algorithms (ATD and LSB).

5.1 LSB Implementation

We have implemented LSB in two different ways. The first way is the traditional LSB with different embedding combinations (224, 242, 422, 134, 143, 314, 341, 413, and 431). It means that, e.g., in combination 143, one-bit is embedded into the Red, four bits into Green, and three bits into the Blue component of the RGB color pixel. The second way is Adaptive LSB (ALSB). In this way, we have two cases:

1) Four bits are embedded into the largest, 3 bits in the middle, and one bit into the lowest color intensity pixel component (ALSBmax).

2) It uses the inverse order of embedding compared to ALSBmin: four bits are embedded into the lowest color intensity pixel component

5.1.1 Traditional LSB with Different Embedding Combinations

In the embedding phase, we create a function for embedding the secret message in the cover image. In this function, we split the cover image into three matrices (planes) R, G, and B. The number of bits embedded in each byte of matrixes (R, G, and B) is determined from the combination that we use. In Appendix D.2, lines 11-22; Line 11 detects how many bits to embed in Blue matrix. Line 21 converts bits of the secret image as binary string to decimal by using the MATLAB function bin2dec. Embedding the secret bits in Blue plane according to (2.1) is made in line 22. In lines 25-34, we embed 4 bits in green as in Blue, and in line 35-46 here embed one bit in Red as in Blue, y, u return the number of pixel touch to embedding the secret message. The full code can be viewed in the Appendix D.2.

The results of traditional LSB with different embedding combinations (224, 242, 422, 134, 143, 314, 341, 413, and 431) are available in Appendix C.9. Then we calculate MSE (2.58), PSNR (2.59), SNR (2.60), WPSNR, MSNR (2.64), and WMSNR as in Appendix D.1, lines 24-44.

In Appendix D.1, line 24, we call a function, MSE, for calculating the MSE as in (2.58). This function receives the number of pixels modified for embedding the secret message cover image, and stego image as input, and calculates the mean square error in Red, Green, and Blue matrixes as in the Appendix D.4, lines 1-16.

In Appendix D.1, line 32, we call function to calculate the value of WAPSNR by calculating the actual maximal value in Red, Green, and Blue matrices. Then calculating APSNR (2.65) for each matrix (Red, Green, and Blue) then weigh them by equal weights (1/3, 1/3, 1/3)) as in Appendix D.7, line 11.

In Appendix D.1, line 25, we calculate the value of MSE over all the image. In line 26, Appendix D.1, we call function, PSNR, to calculate the PSNR as in (2.59) for the image as a whole and each matrix (R, G, and B). Code of PSNR function is given in Appendix D.5, lines 1-6.

In Appendix D.1, line 29, we call the function MSNR to calculate MSNR. In this function, first we calculate the mean value for each matrix (R, G, and B). Then, MSNR is calculated for each matrix (R, G, and B), Appendix D.6, lines 1-14.

In Appendix D.1, lines 40-43, we calculate value of SNR as in (2.60) by calling the function, SNR, Appendix D.8, lines 1-12, for calculating the variances of the color component (R, G, and B) of the cover image and variances of added noise in color component (R, G, and B).

As a sample output of our implementation, the results for embedding the secret message in the cover image 'Lena', when using combination 134 are shown in Figures 5.1 and 5.2.

Cover image Stego image

Figure 5.1: Lena Cover Image and Stego Image for LSB with Combination (134) Implementation. Appendix D.9

```
==========================================================================
cover_image        Size_secret_data            PSNR              MSE
                                                dB
==========================================================================
lena_color            1280000                36.058180         16.116190
PSNR_R=51.141538            PSNR_G=37.946714                PSNR_B=32.400274
MSNR_R=48.122408            MSNR_G=29.734115                MSNR_B=24.738896
MSNR=30.088409
Actual_PSNR_weight(1/3_1/3_1/3)=39.859666
W_PSNR(0.4_0.243_0.357)=41.244564
W_PSNR(0.4_0.3_0.3)=41.560711
W_PSNR(1/3_1/3_1/3)=40.496175
W_MSNR(0.4_0.243_0.357)=35.306139
W_MSNR(0.4_0.3_0.3)=35.590866
W_MSNR(1/3_1/3_1/3)=34.198473
SNR=17.336085
==========================================================================
```

Figure 5.2: Quality Measures for Lena Image After Secret Message Embedding by LSB for Combination (134)

In the extraction function, Appendix D.3, we need combination that is used in embedding phase to extract the secret message.

5.1.2 Adaptive LSB (ALSB) Implementation

In Adaptive LSB (see Appendix E) all of the functions are similar to the traditional LSB except for two differences. The first difference is that we calculate the color intensity by calculating the average of each matrix (Red, Green, and Blue matrices) according to the next formula (5.1):

$$ACk = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} CI(i,j,k)}{MN}. \qquad (5.1)$$

Where CI is the color matrix (Red, Green, and Blue matrices), k=1, 2, 3 , NM is the number of pixels of the image, and ACk is the average of color for matrix k.

As we described before (Section 5.1, beginning), we have ALSBmax and ALSBmin variants of ALSB.

Appendix E.1, in line 9, 11, and 13, we calculate the mean of matrixes (Red, Green, and Blue). In Appendix E.1, lines 33-56, we compare the averages of the matrixes to detect the embedding capacity in each matrix.

In embedding stage, Appendix E.2, lines8-10, we embed the number of bits to be embedded into each matrix (Red, Green, and Blue) in first pixel of cover image for the future use in the extraction stage this the second difference.

The results for embedding the secret message in the cover image ‘Lena’ are shown in Figures 5.1 and 5.3.

```
==========================================================================
cover_image        Size_secret_data           PSNR              MSE
                                               dB
==========================================================================
lena_color           1280000                  35.558127         18.082883
avg_R=60.043004             avg_G=33.020903               avg_B=35.184392
PSNR_R=31.601155             PSNR_G=51.158319               PSNR_B=38.697698
MSNR_R=28.582025             MSNR_G=42.945720               MSNR_B=31.036320
MSNR=29.588356
Actual_PSNR_weight(1/3_1/3_1/3)=39.849215
W_PSNR(0.4_0.243_0.357)=38.887012
W_PSNR(0.4_0.3_0.3)=39.597267
W_PSNR(1/3_1/3_1/3)=40.485724
W_MSNR(0.4_0.243_0.357)=32.948586
W_MSNR(0.4_0.3_0.3)=33.627422
W_MSNR(1/3_1/3_1/3)=34.188022
SNR=15.398574
==========================================================================
```

Figure 5.3: Quality Measures for ALSBmin for Lena Image

For extraction, in main program, Appendix E, we call the extraction function. In this function, Appendix E.3, lines 8-10, we extract from the first pixel of the stego-image according (2.2) how many bits are embedded in each matrix (R, G, and B), and then extract the secret message (see Appendix E.3).

5.2 ATD Implementation

We have implemented ATD with combinations (112,121,211) defining number of ternary digits embedded in each component, (R, G, B) of a color pixel. Also in LSB, we have also considered combinations corresponding to 8 BPP. In this algorithm ATD, we calculate the MSE (2.58), PSNR (2.59), WPSNR, SNR (2.60), MSNR (2.64), and WMSNR only for modified pixel.

In the embedding and extraction phases, given in Appendices F.2, F.3, all of the functions are similar to ATD in gray scale image. Just here we have three matrixes (Red, Green, and Blue), and a number of ternary digits embedded in each matrix depend on the combination we use (112, 121, and 211). The all code of ATD is given in Appendix F.

Results of embedding of the secret message in the cover image ‘Lena’ for the combination (112) are shown in Figures 5.1 and 5.4.

```
===========================================================================
cover_image        Size_secret_data          PSNR             MSE
                                              dB
===========================================================================
lena_color.jpg            1280000          37.550079          11.430689
PSNR_R=37.353306             PSNR_G=37.651402              PSNR_B=37.652425
MSNR_R=34.334176             MSNR_G=29.438803              MSNR_B=29.991048
MSNR=31.580308
W_PSNR(0.4_0.243_0.357)=37.532529
W_PSNR(0.4_0.3_0.3)=37.532471
W_PSNR(1/3_1/3_1/3)=37.552378
W_MSNR(0.4_0.243_0.357)=31.594103
W_MSNR(0.4_0.3_0.3)=31.562625
W_MSNR(1/3_1/3_1/3)=31.254675
Actual_PSNR_weight(1/3_1/3_1/3)=36.915868
SNR=16.354770
===========================================================================
```

Figure 5.4: ATD Quality Measures for Lena Cover Image

5.3 Summary of Chapter 5

Thus, in this chapter, we have implemented and explained ATD and LSB in two difference ways (Traditional LSB with combinations, ALSB), we have presented ‘Lena’ image result as an example. All results and full codes and all functions in ATD and LSB are available in the Appendix D, E, and F.

Chapter 6

EXPERIMENTS RESULTS

In this chapter, we discuss the experiments results of the algorithms for the gray scale (Section 6.1) and color scale (Section 6.2).

6.1 Gray Scale Image Results for ATD, LSBCT, and LSBDT

Fifteen gray scale cover images with size 512×512 pixels used in the experiments are shown in Figure 6.1. Secret messages in these experiments are generated randomly with different sizes. First, we start with the size of a secret message equal to 524288 (512 * 512 * 2) bits, and then increase the size by 30,000 bits every iteration, and record the values of quality measures. The simulation steps show in Figure 6.2.

Figure 6.1: Gray Scale Cover Images Used in ATD, LSBCT, and LSBDT Simulation

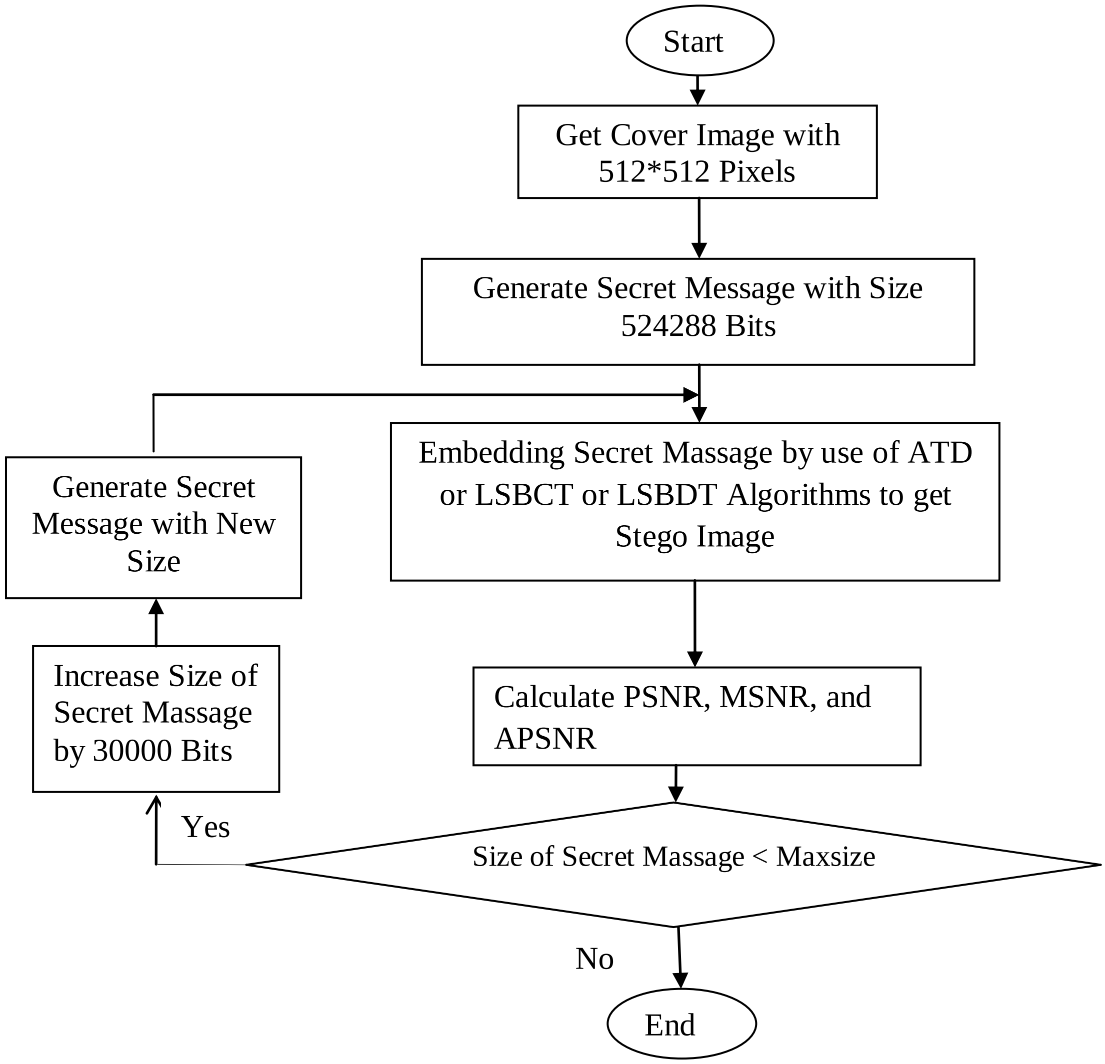


Figure 6.2: Flowchart for Gray Scale Simulation Steps

6.1.1 ATD Results

Table 6.1 shows the PSNR for 15 cover images in Figure 6.3 and EC in the range [2, 3.14] BPP; raw data are in Appendix A.6.

Table 6.1: PSNR of ATD

Cover image	EC (BPP)						
	2	2.22	2.46	2.69	2.80	2.92	3.14
	PSNR (dB)						
1	39.86	38.86	38.41	38.02	37.85	37.69	37.39
2	39.32	38.86	38.44	38.04	37.87	37.69	37.41
3	39.34	38.86	38.43	38.04	37.87	37.69	37.38
4	39.33	38.87	38.42	38.05	37.85	37.71	37.40
5	39.35	38.87	38.44	38.06	37.86	37.69	37.38
6	39.33	38.86	38.44	38.06	37.86	37.68	37.41
7	39.32	38.87	38.44	38.05	37.90	37.69	37.41
8	39.34	38.87	38.45	38.05	37.87	37.68	37.41
9	39.35	38.86	38.44	38.06	37.87	37.69	37.40
10	39.25	38.79	38.37	37.99	37.82	37.63	37.35
11	39.33	38.86	38.44	38.04	37.88	37.69	37.40
12	39.33	38.87	38.42	38.05	37.86	38.68	37.39
13	39.33	38.85	38.44	38.05	37.86	37.70	37.41
14	39.25	38.70	38.27	37.81	37.59	37.41	37.12
15	39.29	38.84	38.40	38.01	37.83	37.67	37.36
Average PSNR (dB)	39.35	38.85	38.42	38.03	37.84	37.73	37.57

For 15 different cover images, we get close results for PSNR. Averaged over 15 images dependence on *EC* is shown in Figure 6.3.

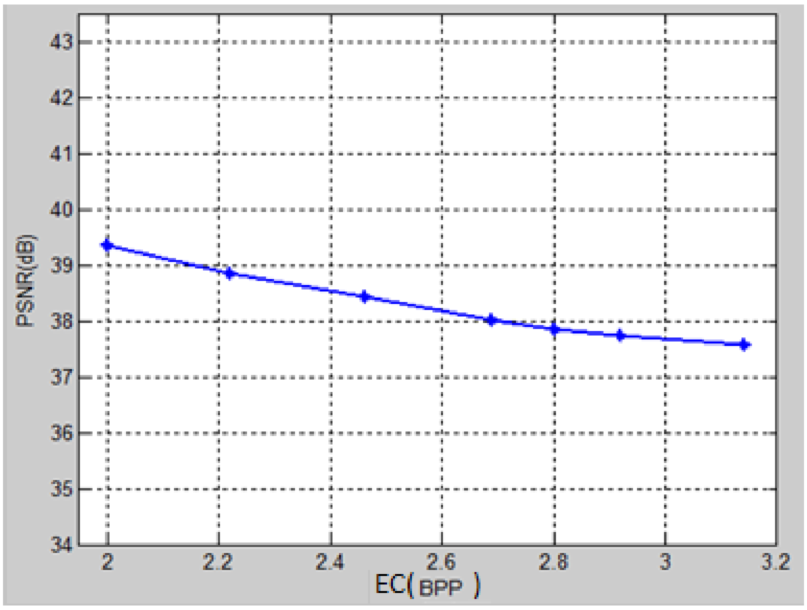


Figure 6.3: Average PSNR of ATD

From Table 6.1 and Figure 6.3, we see that PSNR slightly decreases from 39.35 dB to 37.57 dB when the embedding capacity, *EC*, increases from 2 BPP to 3.14 BPP.

6.1.2 LSBCT Results

We have implemented this algorithm with constant threshold *T*=160 as in [26], and we use two different groups of moduli ((ml=4, mu=8) and (ml=8, mu=16)). These moduli values are defined according to the size of the secret message. Under assumption that the cover image pixel values are distributed uniformly in {0,.., 255}, maximal embedding capacity, *MAX-EC*, dependence on threshold ml, and mu, $MAX - EC\ (T, ml, mu)$, may be defined as follows:

$$MAX - EC(T, ml, mu) = \frac{T}{256} * \log_2 ml + \frac{256-T}{256} * \log_2 mu. \quad (6.1)$$

From (6.1),

$$MAX - EC(160, 4, 8) = \frac{160}{256} * 2 + (256 - 160)/256 * 3 = 2.376\ BPP. \quad (6.2)$$

From (6.2), we see that the secret message for $EC < 2.376$ may be embedded with $T = 160, ml = 4,\ mu = 8$ (as in the first two columns of Table 6.2), but for EC>2.376, it is necessary using $ml = 8, mu = 16$.

Actually, $MAX - EC(160, 8, 16) = \frac{160}{256} * 3 + 96/256 * 4 = 3.375\ BPP$

and all the *EC* in Table 6.2 are less than that value.

Table 6.2: PSNR for LSBCT Algorithm

Cover image	EC (BPP)						
	2	2.22	2.46	2.69	2.80	2.92	3.14
	ml						
	4	4	8	8	8	8	8
	mu						
	8	8	16	16	16	16	16
	PSNR (dB)						
1	42.18	41.75	36.64	36.28	36.11	35.95	35.65
2	41.93	41.40	36.38	36.06	35.91	35.75	35.34
3	42.96	42.70	37.47	37.11	36.96	36.07	36.04
4	42.60	42.40	37.20	36.83	36.71	36.06	36.37
5	41.81	41.28	36.54	36.09	35.90	35.67	35.31
6	41.25	40.90	35.61	35.32	35.20	35.08	34.91
7	41.97	41.48	36.54	36.16	35.99	35.77	35.49
8	40.21	39.80	34.54	34.22	34.06	33.93	33.57
9	40.82	40.41	35.12	34.79	34.65	34.50	34.20
10	40.75	40.49	34.94	34.67	34.58	34.51	34.31
11	42.55	42.11	37.31	36.80	36.61	36.39	36.03
12	41.30	41.03	35.63	35.41	35.35	35.22	35.04
13	42.79	42.36	37.17	36.89	36.74	36.61	36.23
14	41.16	40.77	35.44	35.14	34.95	34.80	34.60
15	43.78	43.73	38.72	38.31	38.12	37.92	37.92
Average PSNR dB	41.87	41.51	36.35	36.01	35.86	35.70	35.41

Table 6.2 shows PSNR for LSBCT for 15 images and values of ml and mu we use in our experiment and BPP $\in [2, 3.14]$. To evaluate the performance of LSBCT, we average PSNR overall the stego images, also shown in Figure 6.4. Raw results are given in Appendix B.4.

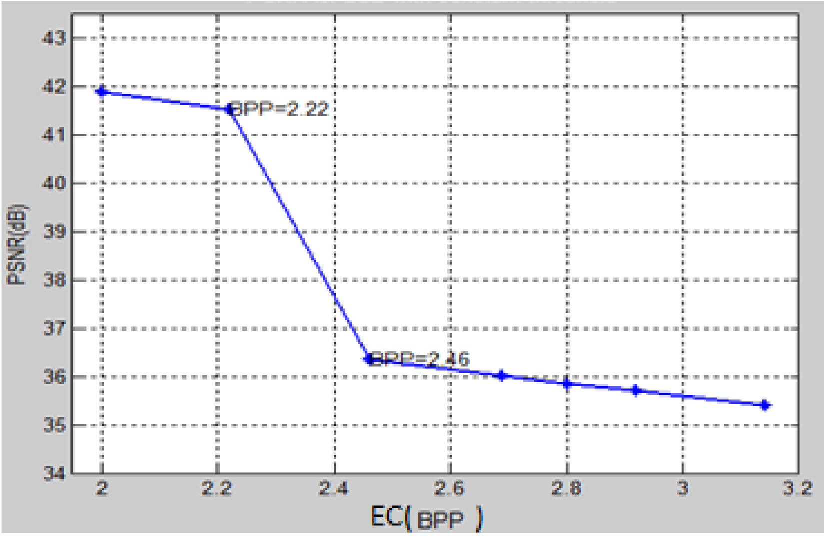


Figure 6.4: Average PSNR for LSBCT

From Figure 6.4, we see that the maximum value of PSNR is 41.87 dB for embedding capacity 2 BPP. On the other hand, when we increase the embedding capacity, we get the minimal value of PSNR about 35.41 for embedding capacity 3.14 BPP. Also, from the Figure 6.4, we see a sharp drop in the value of PSNR when the embedding capacity is higher than 2.22 BPP, which drops from 41.87 dB for 2.22 BPP to 36.35 dB for 2.46 BPP. To avoid the sharp drop as we mentioned before, we use the ml=8 and mu=16 instead of ml=4, mu=8 according to (6.1)-(6.3). This change leads to a great number of bits embedded and, hence, a great distortion.

6.1.3 LSBDT Results

We have implemented LSB with dynamic threshold. The value of the threshold is calculated according to (4.1). Also, here we use two different groups of moduli ((ml=4, mu=8) and (ml=8, mu=16)). These moduli values are defined according to the size of the secret message as in Section 6.1.2, All results are shown in Appendix C.2.

➢ Example of the threshold value calculation:

From Table 6.3, if the size of the secret message is 764288 bits, the maximum embedding capacity in our experiment is 3.14 BPP, and the size of the cover image is 512*512, then we first calculate the embedding capacity, x, according to (2.63).

$$\mathrm{x} = \frac{764288}{512*512} = 2.92.$$

Then substitute x in (4.1):

$$T = \lfloor 160 * (\lceil 3.14 \rceil - 2.92) + 10 \rfloor = 45.$$

Check if value of the threshold, T is a multiple of mu ($mu = 16$) or not. If it is a multiple of mu, then keep it. Otherwise, choose a small number that is a multiple of mu. In this example, result is 45 that is not a multiple of $mu = 16$. Hence, we select a small number that is a multiple of mu=16, it is 32. After that, we calculate the maximal embedding capacity, $MAX - EC,$ according to (6.1) as follows:

$$MAX - EC(32,4,8) = \frac{32}{256} * 2 + \frac{256 - 32}{256} * 3 = 2.875$$

Since $Max - EC(32,4,8) = 2.875 < x = 2.92$ then we can’t embed all secret message with this threshold. Thus, again decrease threshold by mu and recheck the actual embedding capacity in our example $T = 32$ after subtraction $T = 32 - 16 = 16$ then recheck the as follows:

$MAX - EC(16,4,8) = \frac{16}{256} * 2 + \frac{256-16}{256} * 3 = 2.9375, MAX - EC = 2.9375 > x = 2.92$, hence we can embed all secret messages with this threshold, $T = 16$.

Table 6.3: PSNR for LSBDT Algorithm

Cover image	Size of secret message (Bits)						
	524288	584288	644288	704288	734288	764288	824288
	EC(BPP)						
	2	2.22	2.46	2.69	2.80	2.92	3.14
	ml						
	4	4	4	4	4	4	8
	mu						
	8	8	8	8	8	8	16
	T						
	192	160	128	80	48	16	160
	PSNR (dB)						
1	43.21	41.76	40.04	39.03	38.55	38.05	35.60
2	43.42	41.41	40.15	39.29	38.54	38.19	35.36
3	44.12	42.68	40.59	39.33	38.60	38.08	35.47
4	43.18	42.41	41.22	39.48	38.55	38.08	35.37
5	42.94	41.31	39.80	38.34	38.34	38.04	35.33
6	42	40.90	40.09	38.88	38.88	38.52	34.79
7	43.67	41.47	40	38.87	38.36	38.07	35.45
8	40.57	39.79	39.20	38.63	38.31	38.09	35.59
9	41.82	40.41	39.41	38.79	38.41	38.13	34.21
10	43.66	40.50	39.46	38.86	38.54	38.15	34.30
11	43.26	42.11	41.12	39.64	39.05	38.52	35.99
12	43.02	41.02	39.73	39.01	38.80	38.66	35.02
13	43.78	43.74	42.90	42.12	40.06	38.60	34.78
14	42.09	40.78	40.01	39.38	38.97	38.71	34.59
15	43.76	42.36	40.14	38.95	38.48	38.13	35.23
Average PSNR dB	42.97	41.51	40.26	39.30	38.70	38.27	35.48

Table 6.3 shows PSNR of LSBDT algorithm for all images, as well as average over all images, and values of T, ml, and mu that we use in our experiments. As we see from the Table 6.3, the quality of a stego-image drops when the size of the secret message increases; raw results are shown in Appendix C.2.

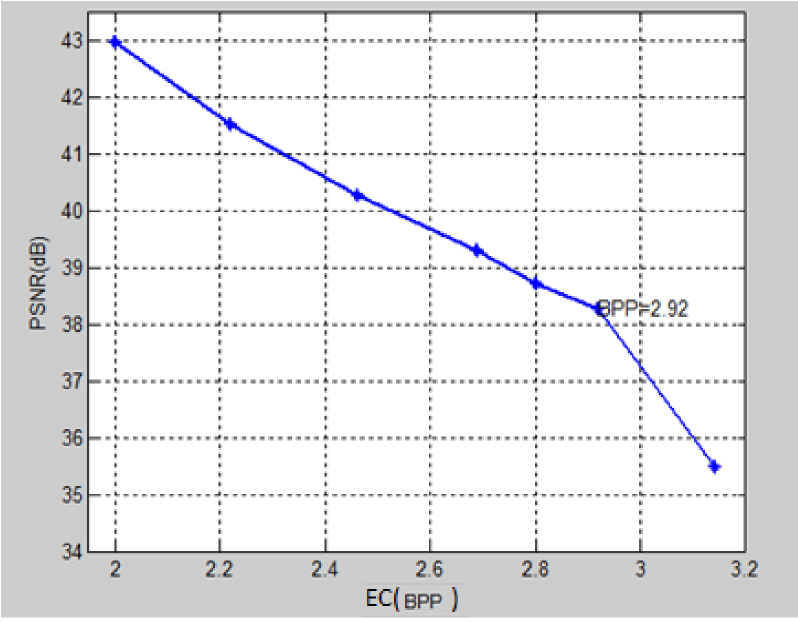


Figure 6.5: Average PSNR for LSBDT

From Figure 6.5, we can see that the maximum value of PSNR it reaches 42.97 dB when the embedding capacity is equal to 2 BPP. Figure 6.5 illustrates a noticeable drop in the value of PSNR when the embedding capacity is between 2 BPP and 2.92 BPP which drops from 42.97 dB to 38.27 dB. Then we see a sharp drop in the value of PSNR when the embedding capacity is higher than 2.92 BPP. PSNR drops rapidly from 38.27 dB for 2.92 BPP to 35.48 dB for EC=3.14 BPP, because the number of distorted bits increases from {2, 3} for ml=4, mu=8 to {3, 4} for ml=8, mu=16.

6.1.4 ATD, LSBDT, and LSBCT Comparison Results

Comparison results for the both methods obtained for PSNR (2.59) are shown in Table 6.4.

Table 6.4: PSNR (dB) for LSBCT (*T*=160), LSBDT, and ATD Dependence on *EC* (BPP)

EC (BPP)	LSBCT (T=160)	LSBDT	ATD
2	41.87	42.97	39.35
2.22	41.51	41.51	38.85
2.46	36.35	40.26	38.42
2.69	36.01	39.30	38.03
2.80	35.70	38.27	37.73
2.92	35.41	35.48	37.57
3.14	41.87	42.97	39.35

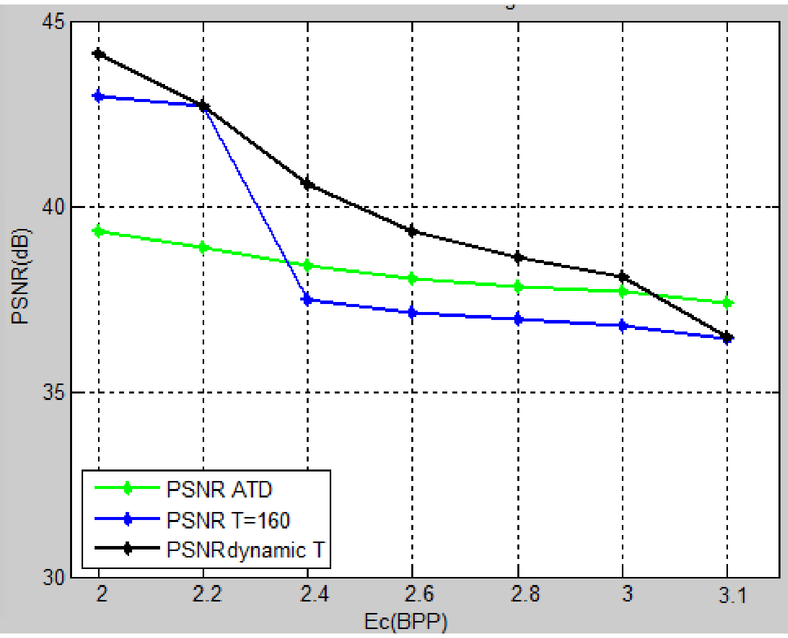


Figure 6.6: PSNR (dB) for ATD and LSBCT (T=160) and LSBDT Dependence on EC (BPP)

From the Table 6.4 and Figure 6.6, it is evident that the quality of the image for ATD is slightly affected by the increase of the embedding capacity. LSBCT and LSBDT have a high quality for the embedding capacity up to 2.22 BPP. LSBDT PSNR reaches 42.97 dB, while LSBCT reaches 41.87 dB for embedding capacity *EC= 2* BPP. For embedding capacity between 2.22 BPP and 2.46 BPP, LSBCT PSNR drops quickly to 36.35 dB. After that, it falls slightly and reaches 35.41 dB at *EC=3.14* BPP.

ATD has a higher PSNR than LSBDT when embedding capacity is more than 3 BPP; this is because in LSBDT uses modules with values {8, 16} for the large-size secret messages. It means that in each pixel it embeds 3 or 4 bits according to the threshold, and the maximum possible distortion by embedding is 15, and for ATD, the maximum possible distortion is 10 (e.g., when sub1 and sub2 are increased from 0 to 1, and then sub1 and resulting pixel are again increased by 1; each increase of sub1, gives increase by 4, an increase of sub2 and of the pixel results in increase by 2).

6.1.5 Comparison Versus Known Experiments Results

From the known experiments are shown in section 2.6, Table 2.1, Figure 2.17, and from our results in Table 6.4 and Figure 6.6, when we compare the ATD and LSBCT as in the known experiments we don't get the same result as in the paper [28]. Therefore, we modify the LSBCT to LSBDT to get the same result as in the paper [28].

6.2 Results for Color Images

In this section, we discuss the results of LSB, ALSBmax, ALSBmin, and ATD for different combinations of embedding into color scale images.

6.2.1 LSB and ALSBmax, ALSBmin Results

Cover color scale images with sizes of 512×512 pixels used in the experiments are shown in Figure 6.7. The secret message in this experiment is constant. The raw results are shown in Appendices D.9 and E.3. The simulation steps show in Figure 6.8.

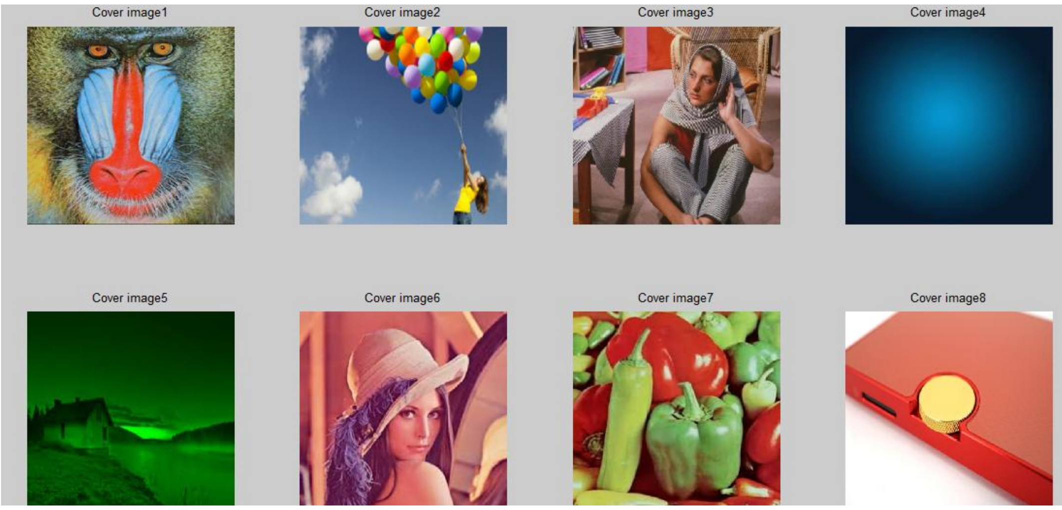


Figure 6.7: Cover Images Used for Experiments with ATD and LSB

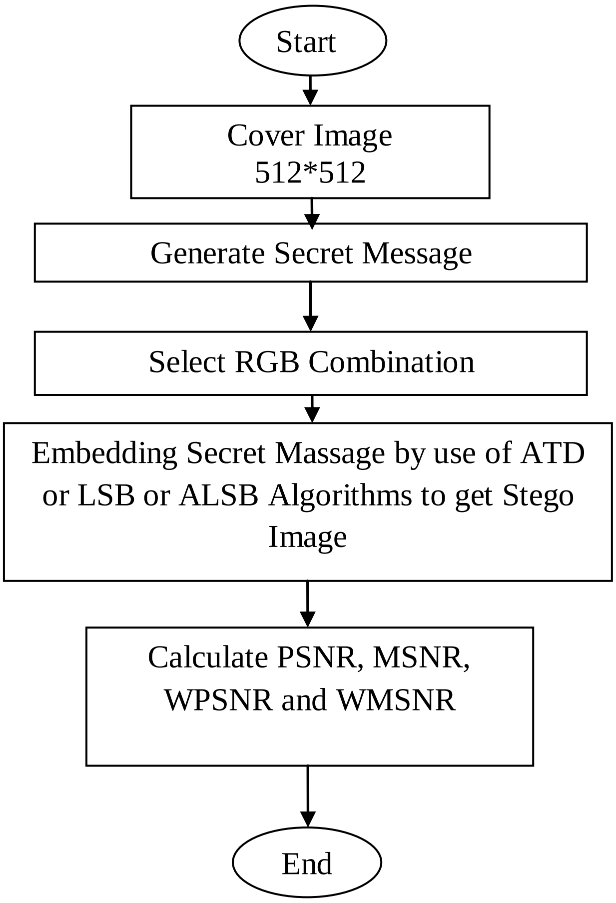


Figure 6.8: Flowchart for Color Image Embedding by ATD, LSB, and ALSB Simulation Steps

Table 6.5 shows the PSNR of LSB for combinations (134, 143, 341, 314, 413, 431, 224, 242, and 422); and PSNR of Adaptive LSB in both cases the raw results are in

Appendices D.9 and E.3.

Table 6.5: PSNR (dB) for LSB in Different Combinations and ALSBmax, ALSBmin

LSB for different combinations										ALSBmax	ALSBmin
Color Image	4_2_2	2_4_2	2_2_4	4_3_1	3_4_1	4_1_3	1_4_3	3_1_4	1_3_4		
Pepper	36.15	33.9	36.97	35.64	35.68	35.75	35.58	36.35	36.46	35.58	36.46
Lena	35.95	36.15	36.63	35.46	35.68	35.57	34.3	36.05	34.6	36.05	34.6
Baboon	36.17	36.13	35.05	37.01	35.66	35.77	35.62	36.32	36.33	36.62	36.33
Barbra	33.16	36.1	34	36.63	35.66	35.76	35.62	36.3	36.31	35.63	36.31
Balloon	36.18	36.16	37.01	35.64	35.66	35.78	35.63	36.4	36.38	36.38	36.4
Blue	38.28	36.19	37.66	37.44	34.52	35.01	35.66	38.89	34.08	34.08	37.44
Green	34.46	36.39	36.85	34.1	35.58	34.24	33.08	35.99	36.29	34.1	34.1
Red	35.47	36.09	33.51	34.98	35.7	36.5	35.55	35.9	35.91	34.98	35.91

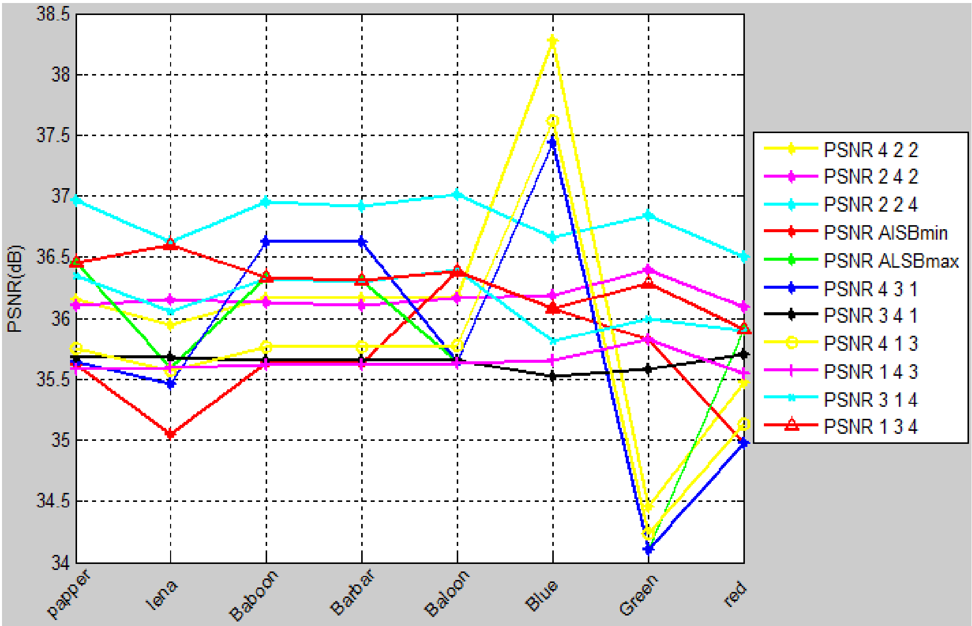


Figure 6.9: PSNR for LSB for Different Embedding Combinations and ALSBmax, ALSBmin

As we can see from the Table 6.5 and Figure 6.9, different LSB combinations for various images are in different relations while the embedding capacity is the same. EC=8 BPP. For example, for Lena, PSNR=35.95 dB in 422 combination is less than PSNR=36.15 dB for 242 combination, whereas for Blue, PSNR for 422 combination is greater than PSNR for 242 combination.

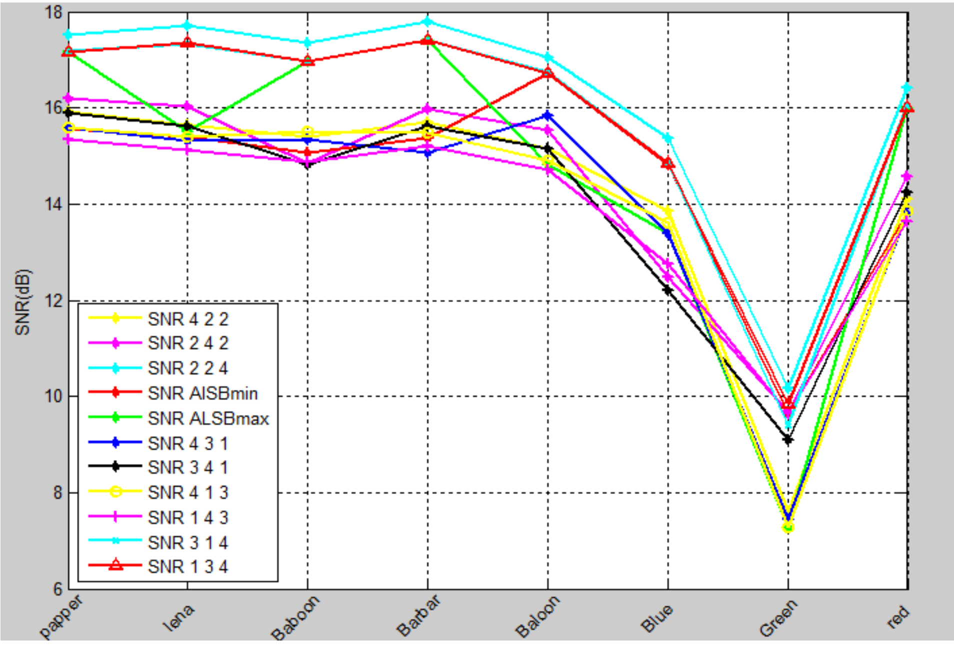


Figure 6.10: SNR for LSB for Different Embedding Combinations and ALSBmax, ALSBmin

From the other side, SNR (2.60) for LSB is less than PSNR dependent on the images but still two combinations on different images may be in inverse relationships (e.g., SNR for 431 on Barbara is less than SNR for 431, but on Balloon, SNR for 431 is greater than SNR for 431) as we can see in Figure 6.10.

6.2.2 ATD Results

In this algorithm as in gray scale, we convert the secret message to base 3, and we embed in each pixel 4 ternary digits in different combinations 211, 121, and 112. For example, if we use 211 combination, it means that two ternary digits are embedded in red color component, one ternary digit in Green color component, and one ternary digit into Blue color component. Results are given in Table 6.6 and Figure 6.11; raw results are in Appendix F.3.

Table 6.6: PSNR (dB) for ATD for Different Combinations Result

Cover Image	Combinations		
	211	121	112
Peppers	37.41	37.42	37.42
Lena	37.55	37.55	37.55
Baboon	37.56	37.56	37.56
Barbara	37.57	37.56	37.57
Balloon	37.42	37.43	37.43
Blue	37.55	37.54	37.54
Green	34.50	34.50	34.50
Red	36.77	36.76	36.77

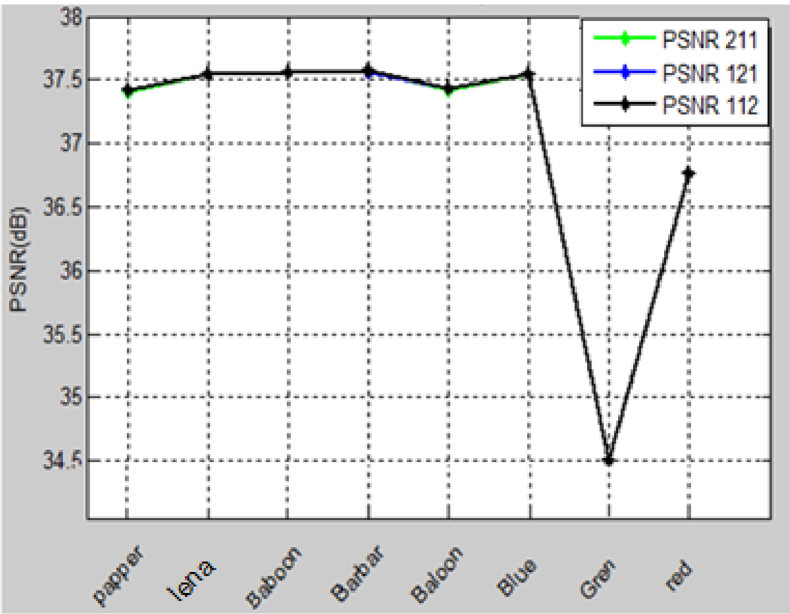


Figure 6.11: PSNR for ATD for Different Embedding Combinations

From Figure 6.11 and Table 6.6, we see that PSNR for ATD is not affected by the combination but it depends on image; PSNR for this algorithm is in range from 34.50 dB to 37.57 dB; we get the minimum value in the Green image for all combinations.

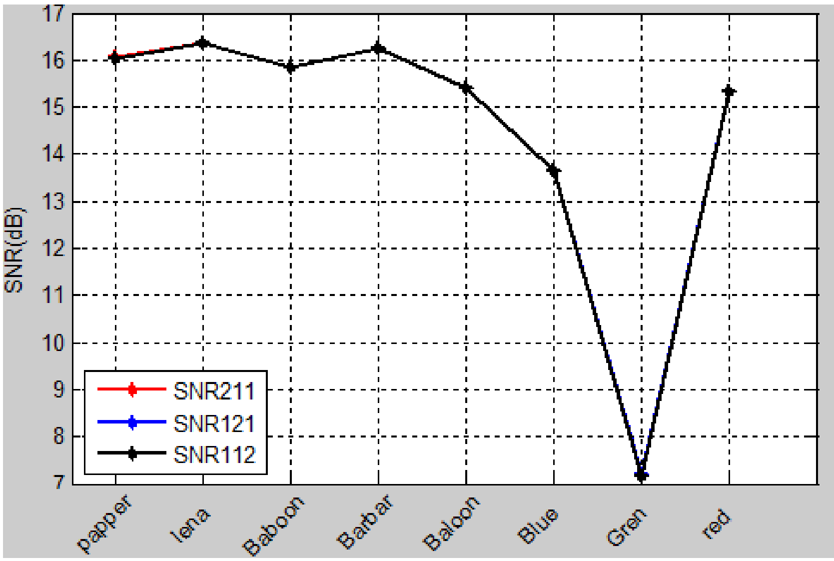


Figure 6.12: SNR for ATD in Different Embedding Combinations

For SNR we get the smallest value for Green image is 7.1 dB, and Lena image has the largest value of SNR, 16.5 dB, as Figure 6.12 shows.

6.2.3 ATD and LSB Comparison Results

Figure 6.13 shows ATD versus LSB comparison results using 512×512 cover images.

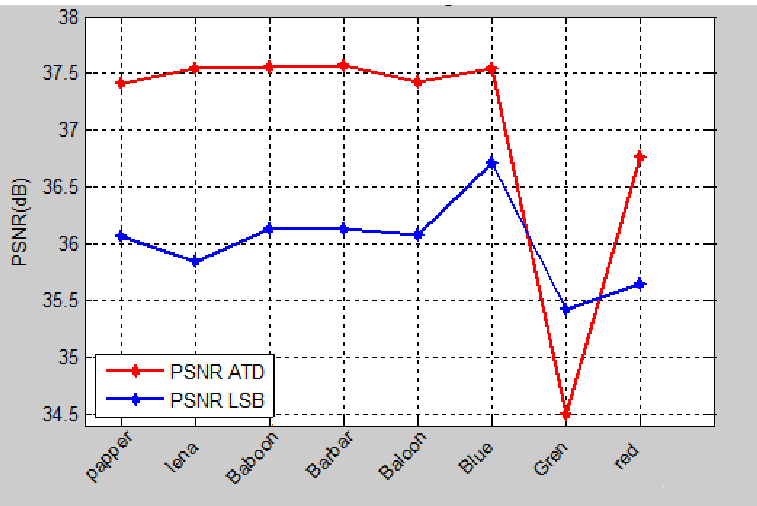


Figure 6.13: PSNR for ATD and LSB

As we can see from Figure 6.13, ATD has a large value of PSNR when we compare it with LSB algorithm, except for the Green image. In this image, the ATD algorithm gets a small value of 34.5 dB. The value of PSNR for LSB algorithm changes between 35.4 dB and 36.7dB.

On the other hand, SNR for ATD and LSB is very close to each other as we see in Figure 6.14 except for the Green image. The quality of ATD is less than LSB as shown in Figure 6.14 which drops to approximately 9 dB for LSB and to 7 dB for ATD.

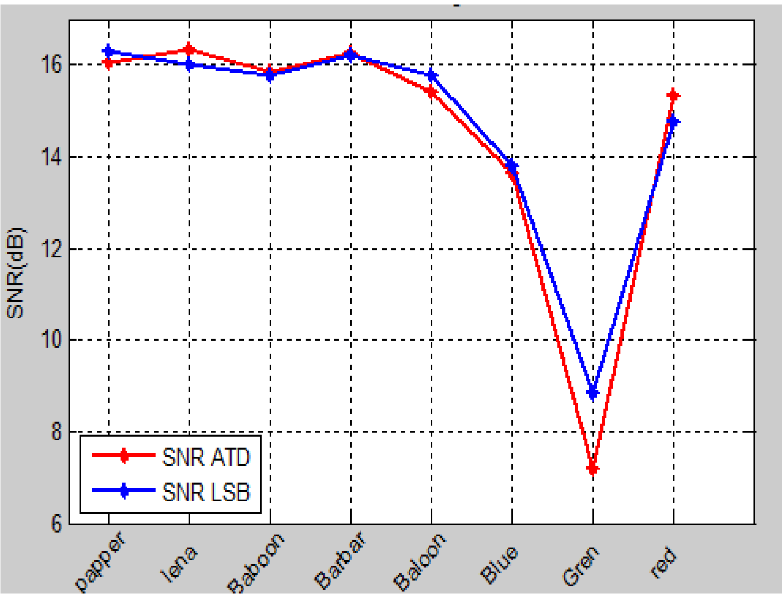


Figure 6.14: SNR (dB) for ATD and LSB

6.2.4 Comparison Versus Known Experiments Results

From the known experiments are shown in section 2.5, Table 2.2, in [24] they implement LSB with combinations 134 on two cover images (Lena, Baboon), in addition, we implement LSB with 8 different combinations and on 8 cover images shown in figure 6.7. Also, we modify LSB to ALSB (ALSBmax, ALSBmin).

6.3 Study of the Quality Metrics

According to our experiments which we discussed in Section 6.2, when we embed 8 bits in each pixel, we obtain PSNR a different behavior for various combinations and images. We expect that good combinations shall depend on image and shall be little dependent on the combinations since all of them modify the same number of bits (8 bits).To solve this problem, we consider the following measures: MSNR (Mean Signal to Noise Ratio) and APSNR (Actual PSNR). MSNR (2.64) is calculated as the mean value of all image pixels and changing the possible peak signal value, which is 255 in equation (2.59), to the mean value of matrix. And APSNR (2.65) improves PSNR by considering actual peak signal instead of possible peak signal which is 255 in equation (2.59). Then we weigh MSNR and PSNR by three different groups of weights (group1values, (0.4, 0.243, and 0.357)), we get as the follows:

From Figure 2.15, we get $h_R = 0.35,\ h_G = 0.3125$, and $h_B = 0.2125$,

$H_{sum} = h_R + h_G + h_B = 0.875$.

After normalizing:

$$h_R^{`} = \frac{h_R}{H_{sum}} = \frac{0.25}{0.875} = 0.4,$$

$$h_G^{`} = \frac{h_G}{H_{sum}} = \frac{0.2125}{0.875} = 0.243,$$

$$h_B^{`} = \frac{h_B}{H_{sum}} = \frac{0.3125}{0.875} = 0.357.$$

We have suggested two other group of weights (group 2 values, (0.4, 0.3, 0.3), and group3 values (1/3, 1/3, 1/3)).

6.3.1 MSNR and APSNR for Gray Scale Images

- ATD Results

APSNR for ATD dependence on embedding capacity is shown in Figure 6.15. APSNR decreases with the increase of the embedding capacity. It arrives at 37.07 dB when embedding capacity is equal to 3.1 BPP.

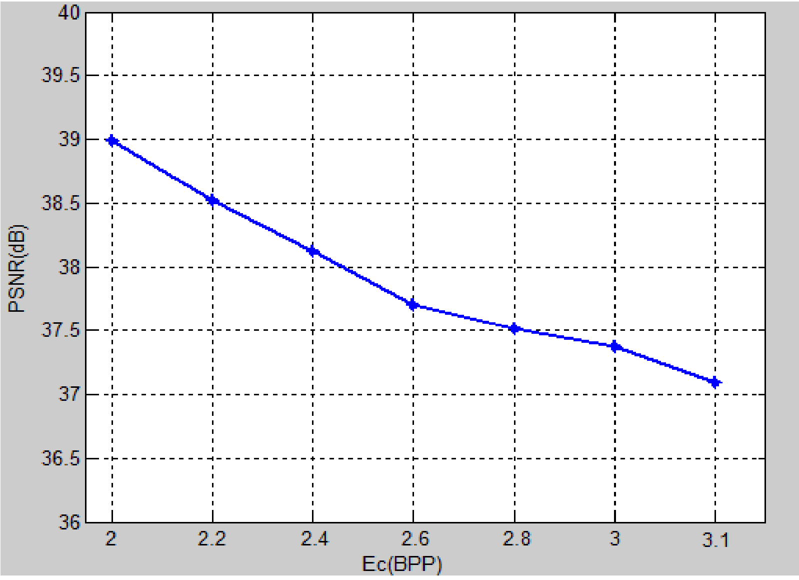


Figure 6.15: APSNR (dB) of ATD

MSNR for ATD dependence on embedding capacity is shown in Figure 6.16. We see that MSNR has the same form as PSNR and APSNR. It decreases from 32.85 dB when *EC*=2 BPP to 30.87 when *EC*=3.1 BPP.

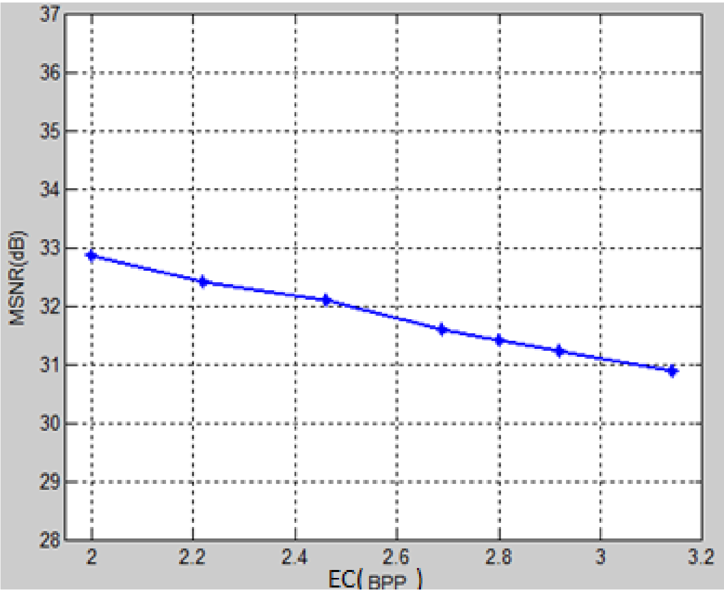


Figure 6.16: MSNR (dB) of ATD

- LSBCT Results

APSNR for LSB with Constant Threshold algorithm reaches the maximum value

41.53 dB for embedding capacity, *EC*= 2 BPP and minimum value 35.07 dB for 3.1 BPP as shown in Figure 6.17. For MSNR there is a severe decrease in the quality of the stego image when increasing the embedding capacity, it drops to 29.10 dB, shown in Figure 6.18.

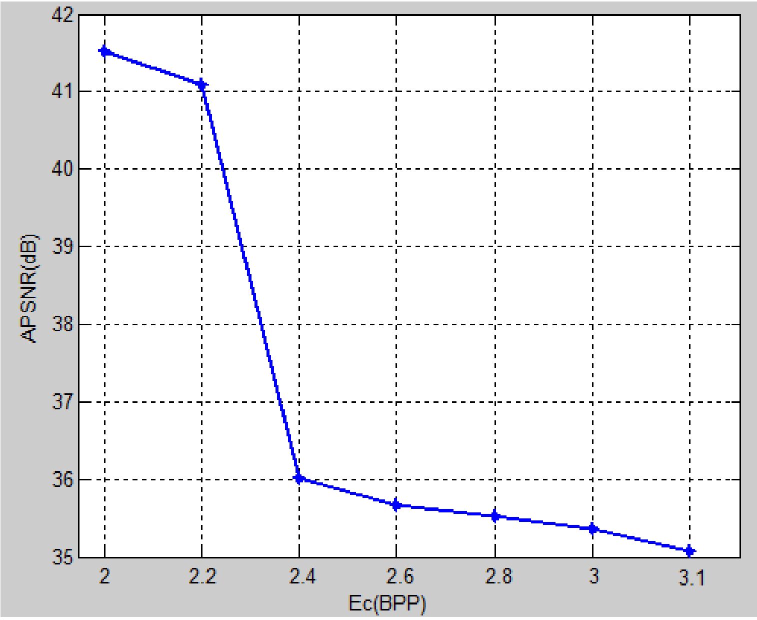


Figure 6.17: APSNR (dB) for LSBCT

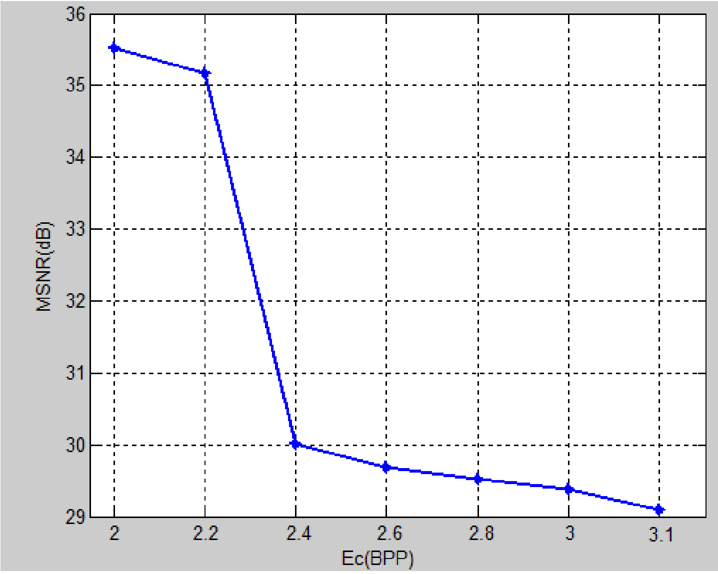


Figure 6.18: MSNR (dB) for LSBCT

- LSBDT Results

APSNR and MSNR for LSBDT in Figures 6.19 and 6.20 reach the maximum value 42.62 dB and 36.58 dB respectively for the embedding capacity, *EC*= 2 BPP. So these values quickly drop to 34.99 dB and 29.09 dB, respectively.

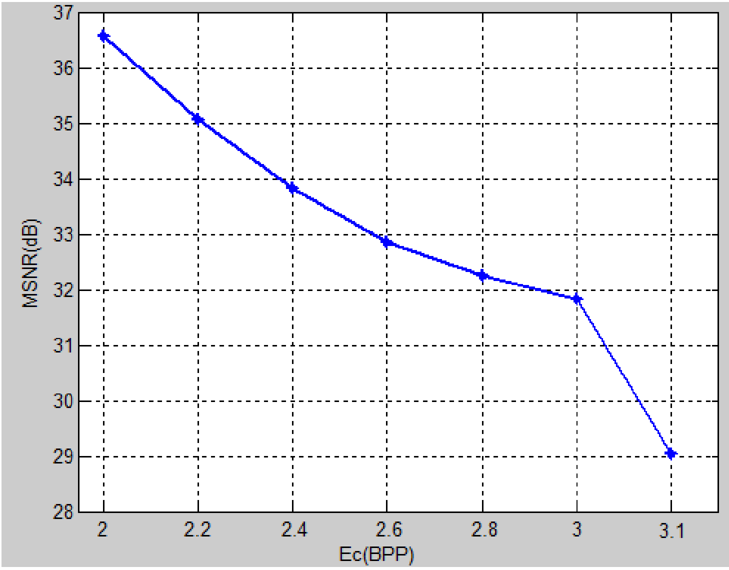


Figure 6.19: APSNR (dB) for LSBDT

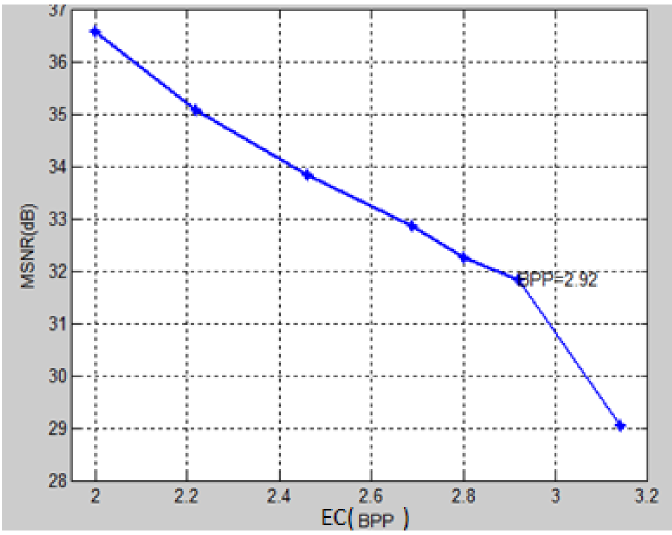


Figure 6.20: MSNR (dB) for LSBDT

- ATD, LSBCT and LSBDT Comparison Using MSNR and APSNR

From Figures 6.21 and 6.22, we can see that MSNR and APSNR for the three algorithms (LSBCT and LSBDT, and ATD), have the same form as of PSNR. LSBDT reaches 42.62 dB for APSNR, and 36.58 dB for MSNR, while LSBCT gets 41.53 dB for APSNR, and 35.52 dB for MSNR when EC=2 BPP. When EC increases from 2.2 BPP to 2.4 BPP, LSBCT quality drops quickly in all metrics it gets 36.01 dB for APSNR, and 30.02 dB for MSNR. After that, it falls slightly to reach 35.07 dB for APSNR, and 29.10 dB for MSNR.

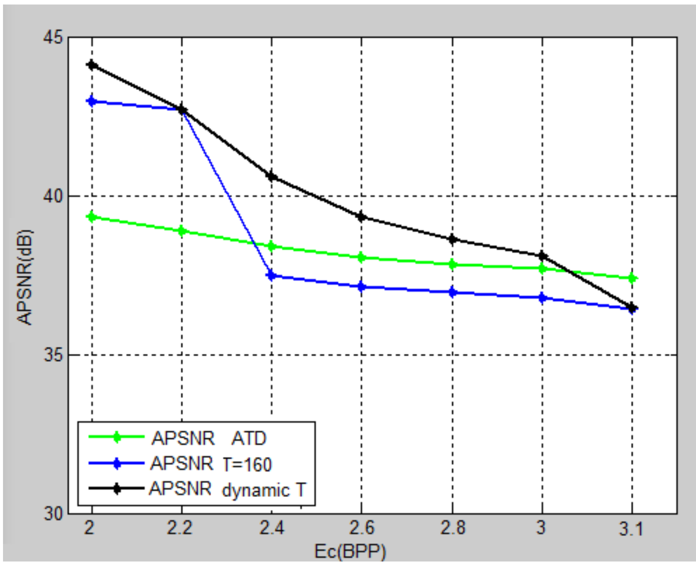


Figure 6.21: APSNR (dB) for ATD and LSBCT and LSBDT

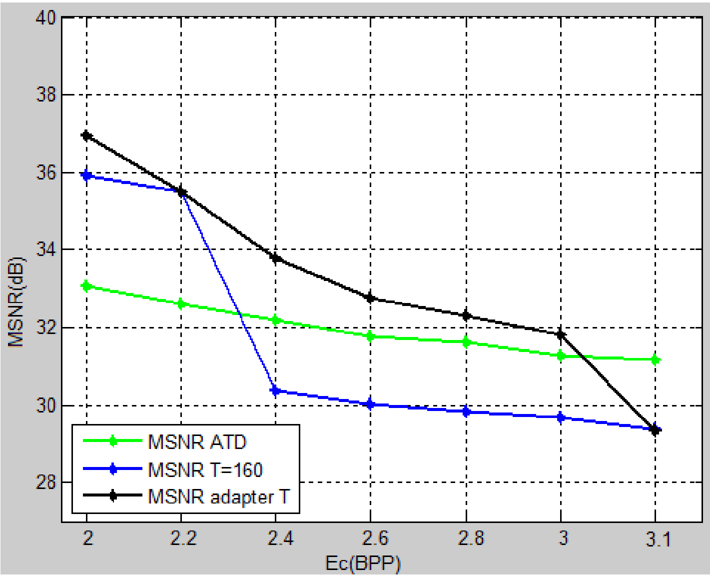


Figure 6.22: MSNR (dB) for ATD and LSBCT and LSBDT

6.3.2 WPSNR, MSNR, and WMSNR for Color Images

- LSB Results

From Figures 6.23-6.24, we see that PSNR after weighing for LSB with embedding capacity 8 BPP and the values of weight (0.4, 0. 243, 0.357) come from human color perception as in Section 6.3 and other weights (0.4, 0. 3, 0.3) and (1/3,1/3,1/3) we take values close to that weight calculate from human color perception, for different combinations still fluctuates but as we see from figures 6.23-6.25 the PSNR with weights (1/3,1/3,1/3), it has more plateau when we compare it with other weights (0.4, 0. 3, 0.3) and (0.4, 0. 243, 0.357), and PSNR.

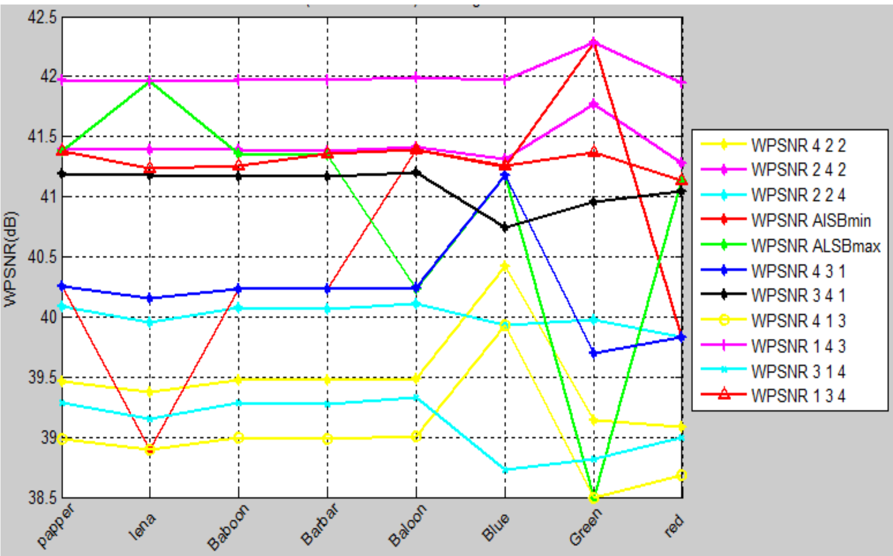


Figure 6.23: Weighed PSNR (dB) by (0.4, 0.243, and 0.357) for LSB for Different Embedding Combinations

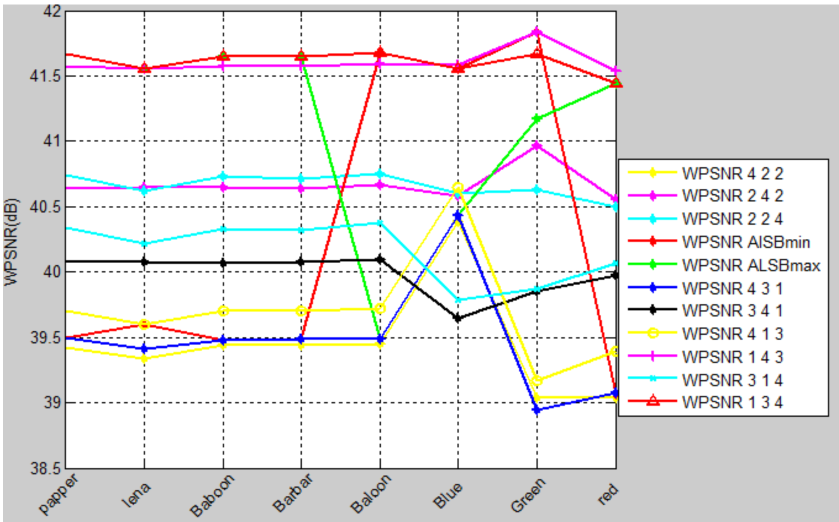


Figure 6.24: Weighed PSNR (dB) by (0.4, 0. 3, 0.3) for LSB for Different Embedding Combinations

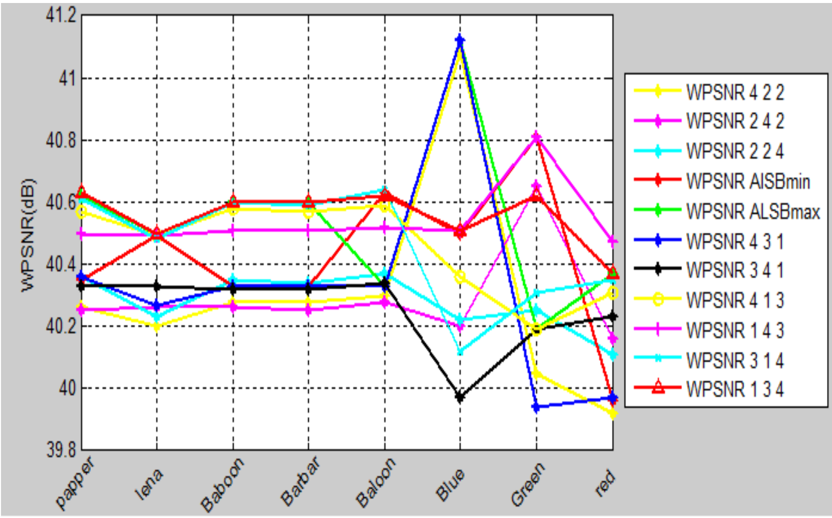


Figure 6.25: Weighed PSNR (dB) by (1/3, 1/3, 1/3) for LSB for Different Embedding Combinations and ALSBmax, ALSBmin

According to our result in PSNR and WPSNR for different weights, we consider MSNR as in Figure 6.26 and Table 6.7 show the value of MSNR for different combinations as we described in Chapter 5. We calculate the mean value of an image (2.64) and this changed the maximum possible value 255 in formula (2.59) by mean value of the image. With MNSR we get improvement in the result obtained by PSNR and WPANR as we show in Figure 6.26, MSNR has nearly one and the same value for different combinations with the same embedding capacity but we see there is more improvement in the result obtained by considering WMSNR with three groups weights as Section 6.3.

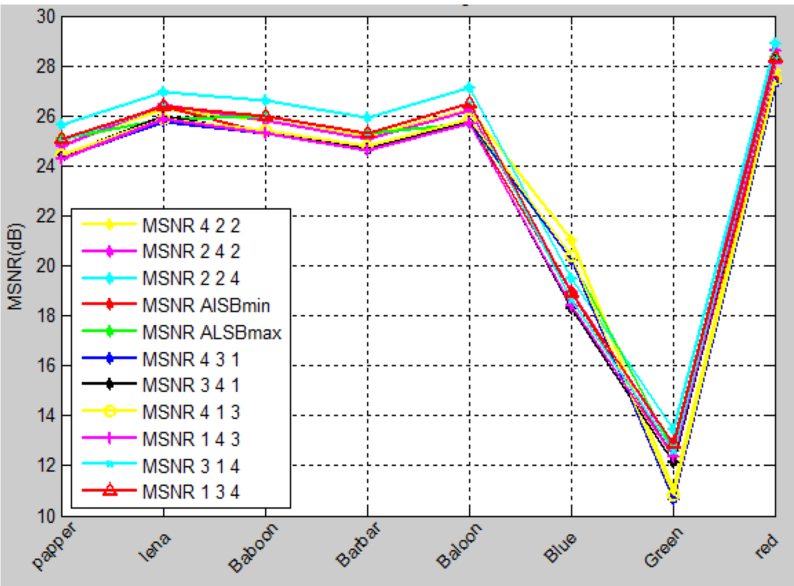


Figure 6.26: MSNR (dB) for LSB for Different Embedding Combinations and ALSBmax, ALSBmin

Table 6.7: MSNR (dB) for LSB and ALSB Methods Result

Cover Image	LSB for different combinations									ALSBmax	ALSBmin
	4_2_2	2_4_2	2_2_4	4_3_1	3_4_1	4_1_3	1_4_3	3_1_4	1_3_4		
Pepper	24.86	24.81	25.66	24.34	24.38	24.46	24.27	25.04	25.05	25.04	25.05
Lena	26.28	26.44	26.98	25.78	25.96	25.89	25.88	26.40	26.41	26.41	25.88
Baboon	25.84	25.79	26.61	25.30	25.32	25.44	25.28	25.99	25.99	25.30	25.99
Barbra	25.16	25.09	25.93	24.62	24.66	24.76	24.60	25.32	25.32	24.63	25.32
Balloon	26.28	26.20	27.12	25.74	25.73	25.89	25.68	26.50	26.49	25.73	25.68
Blue	21.05	18.96	19.5	20.21	18.28	20.38	18.40	18.65	18.92	18.92	20.21
Green	11.04	12.76	13.43	10.66	12.13	10.82	12.34	12.57	12.86	12.34	12.57
Red	27.79	28.60	28.90	27.31	28.20	27.46	28.08	28.28	28.30	27.46	28.30

To improve MSNR, we weigh MSNR similar as we have done for PSNR. Results obtained are shown in Figures 6.26-6.28.

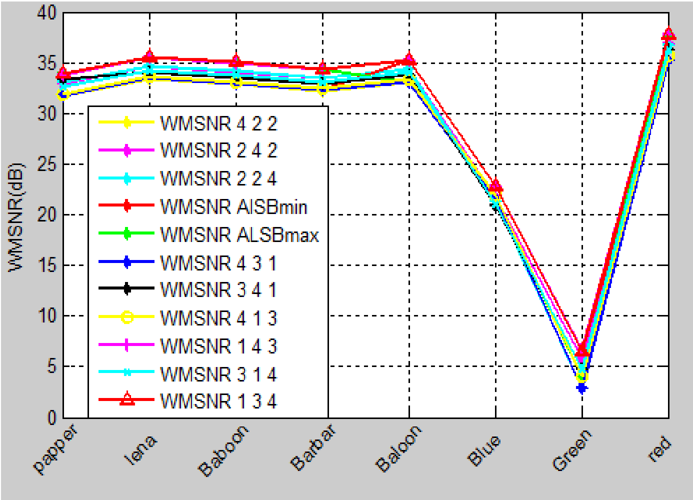


Figure 6.27: Weighed MSNR (dB) by (0.4, 0.3, and 0.3) for LSB for Different Embedding Combinations and ALSBmax, ALSBmin

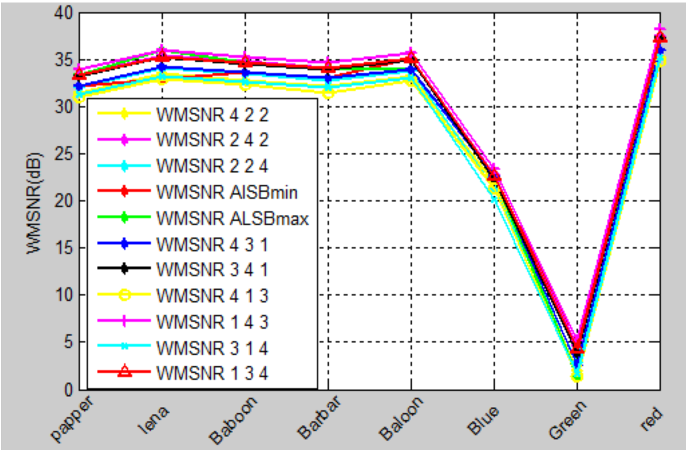


Figure 6.28: Weighed MSNR (dB) by (0.4, 0.243, and 0.357) for LSB for Different Embedding Combinations and ALSBmax, ALSBmin

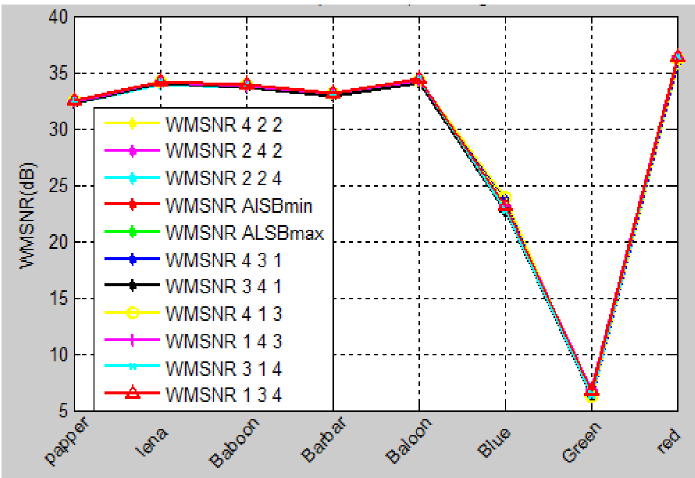


Figure 6.29: Weighed MSNR (dB) by (1/3, 1/3, 1/3) for LSB for Different Embedding Combinations and ALSBmax, ALSBmin

As we see from Figures 6.27-6.29, weighed MSNR with weight (1/3, 1/3, 1/3) looks invariant for different combinations when we compare it with other weights (0.4, 0.3, 0.3) and (0.4, 0.243, 0.357). It behaves as we expect because embedding capacity for different combinations is constant where *EC*=8 BPP, with a different number of bits embedded in Red, Blue, and Green components.

- ATD Results

When we calculate MSNR (2.64) and we weigh MSNR with three different group weights (1/3,1/3,1/3), (0.4, 0. 3, 0.3) and (0.4, 0. 243, 0.357). From Figure 6.30 and Table 6.8, we see that, the value of MSNR and WMSNR in different weights for ATD in different combinations has the same value.

Table 6.8: MSNR (dB) for ATD for Different Combinations

Cover Image	Combinations		
	211	121	112
Peppers	29.82	29.82	29.82
Lena	31.58	31.58	31.58
Baboon	30.93	30.92	30.92
Barbara	30.27	30.26	30.27
Balloon	30.22	31.22	31.22
Blue	24.02	24.01	24.01
Green	14.74	14.75	14.74
Red	32.79	32.97	32.97

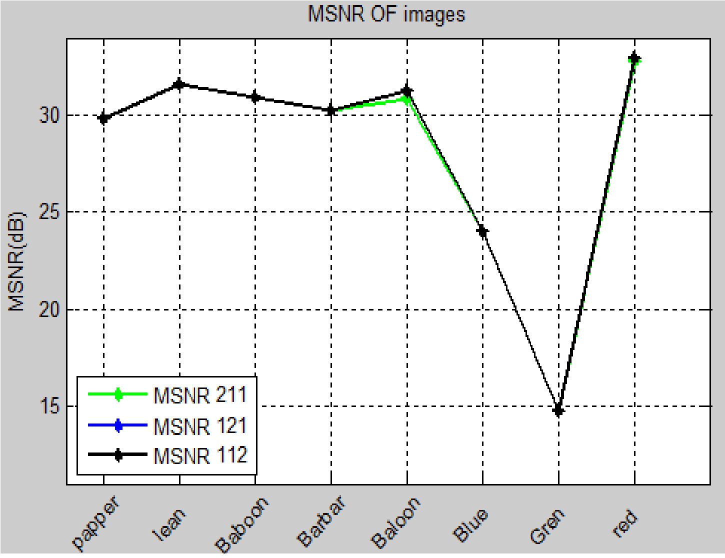


Figure 6.30: MSNR (dB) for ATD for Different Combinations

- ATD and LSB Comparison Results Using WPSNR, MSNR and WMSNR

When we compare LSB and ATD according to WPSNR with different weights, we get that LSB algorithm has the best quality image according to WPSNR. The value of WPSNR for LSB is in between 40.60 dB and 40.20 dB for different weights as shown in Figure 5.31.

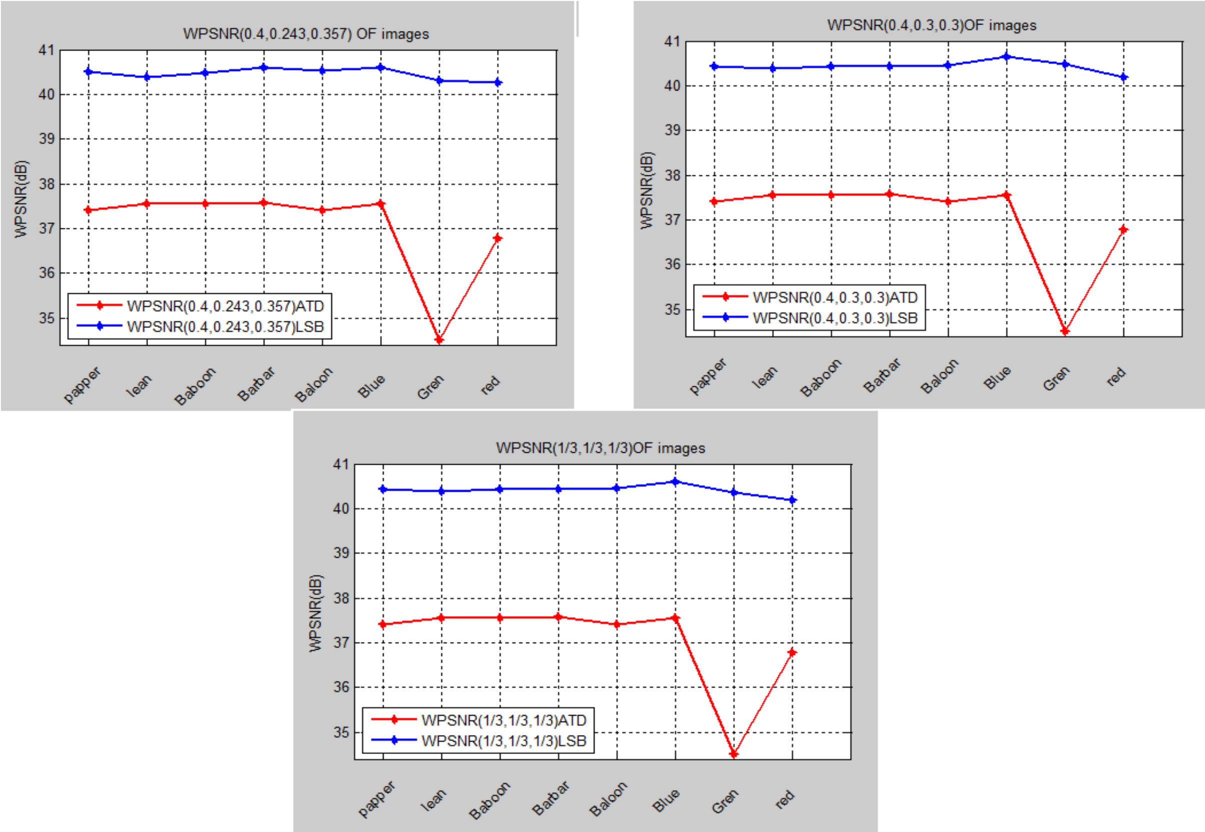


Figure 6.31: WPSNR (dB) for ATD and LSB with Different Weights (0.4, 0.243, 0.357), (0.4, 0.3, 0.3), and (1/3, 1/3, 1/3)

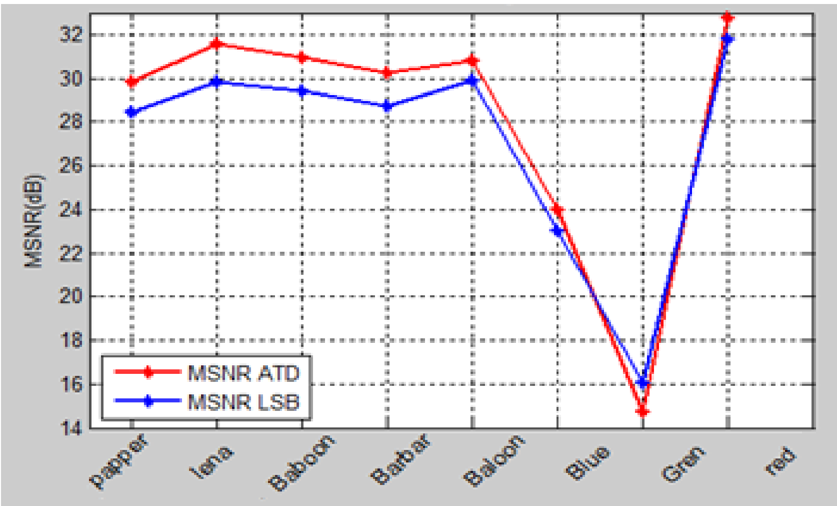


Figure 6.32: MSNR (dB) for ATD and LSB

For MSNR, the both algorithms LSB and ATD have the same curve form (see Figure 6.32). When we compare them, ATD has noticeably larger value than LSB for all images except 'green image' for this image; the LSB has a larger value of 16 dB, while ATD has a value of 14.74 dB. When we compare LSB and ATD for WMSNR, from Figure 6.33, we can see that LSB algorithm has a large value than ATD for all images just if we except Green image; for this image, where ATD has a large value, 15 dB, while LSB has approximately 5 dB.

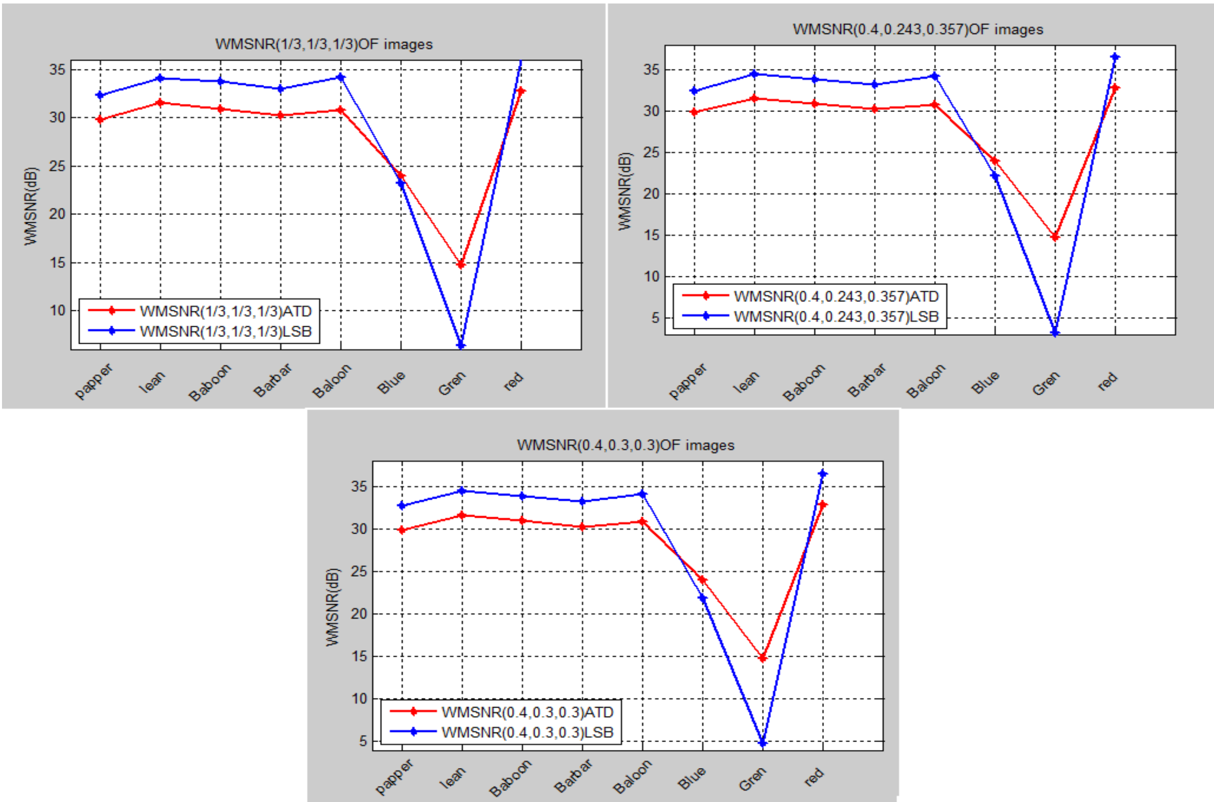


Figure 6.33: WMSNR (dB) for ATD and LSB with Different Weights (0.4, 0.243, 0.357), (0.4, 0.3, 0.3), and (1/3, 1/3, 1/3)

6.3.3. Evaluation of Different Metrics

- Evaluation of Different Metrics for LSB Algorithm

We measure the quality of the stego images in LSB algorithm by different criterion measure PSNR, and MSNR, WMSNR, and WPSNR with weights (1/3,1/3,1/3), (0.4, 0.3, 0.3), and (0.4, 0.243, 0.357). We compare between these metrics by using deviation of the results in each criterion the deviation is calculated according to (6.3):

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{X})^2} \qquad (6.3)$$

Where $\overline{X}$ is the mean value of the pixels of a cover calculated according to (2.62) and N is the number of elements. To compare between these metrics (PSNR, and MSNR, WMSNR, and WPSNR with weights (1/3,1/3,1/3), (0.4, 0.3, 0.3), and (0.4, 0.243, 0.357) we calculate deviation for each cover image in different combinations than average of the deviation over 8 cover images. The results are shown in Table 6.9.

Table 6.9: Deviation of Results in Each Metric for LSB Algorithm

Metric	Deviation
PSNR	0.92
WPSNR (0.4, 0.243, 0.357)	1.07
WPSNR (0.4, 0.3, 0.3)	0.896
WPSNR (1/3,1/3,1/3)	0.768
MSNR	0.663
WPSNR (0.4, 0.243, 0.357)	0.554
WMSNR (0.4, 0.3, 0.3)	0.492
WMSNR (1/3,1/3,1/3)	0.211

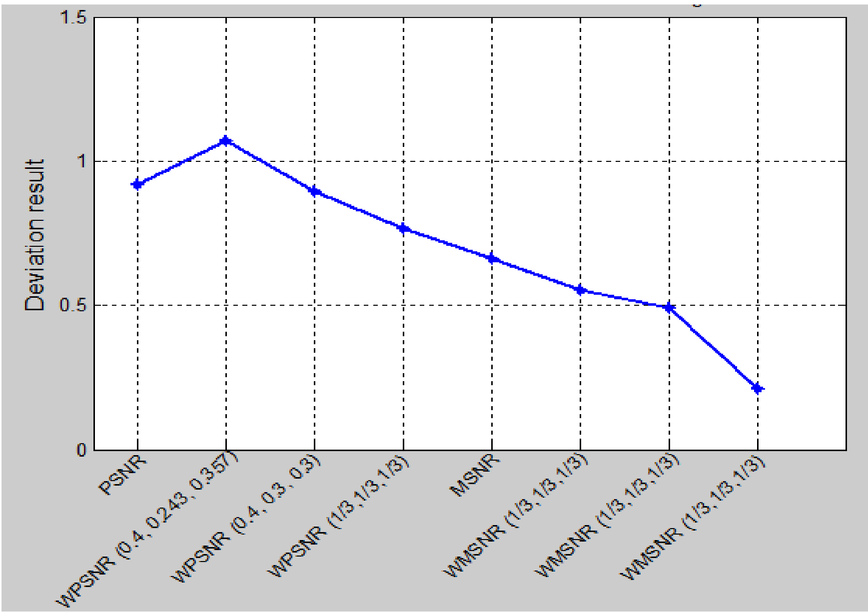


Figure 6.34: Deviation of Results in Each Metric for LSB Algorithm

When we compare result of deviation of the criteria as we show in Figure 6.34 and Table 6.8, the minimal deviation of the metrics for LSB algorithm we get in WMSNR with weight (1/3,1/3,1/3): the value of deviation is 0.211, and maximum result is in WPSNR with weight (0.4, 0.243, 0.357). These weights we get from human color perception in Figure 2.15 as shown at the beginning of Section 6.3. Overall, the best metric for stego images for LSB algorithm is MSNR with weights (1/3, 1/3, 1/3).

- Evaluation of Different Metrics for ATD Algorithm:

We use different criteria to evaluation the quality of the stego images for ATD algorithm as PSNR, WPSNR with weights (1/3,1/3,1/3), (0.4, 0.3, 0.3), and (0.4, 0.243, 0.357), MSNR, and WMSNR with weights (1/3,1/3,1/3), (0.4, 0.3, 0.3), and (0.4, 0.243, 0.357). Also for this algorithm, we compare between these criteria measures by calculating deviation (6.1) of result in each criterion measure for ATD algorithm.

Table 6.10: Deviation of Results with Different Metrics for ATD Algorithm

Metric	Deviation
PSNR	0.0040
WPSNR (0.4, 0.243, 0.357)	0.0039
WPSNR (0.4, 0.3, 0.3)	0.0035
WPSNR (1/3,1/3,1/3)	0.0042
MSNR	0.088
WPSNR (0.4, 0.243, 0.357)	0.086
WMSNR (0.4, 0.3, 0.3)	0.079
WMSNR (1/3,1/3,1/3)	0.088

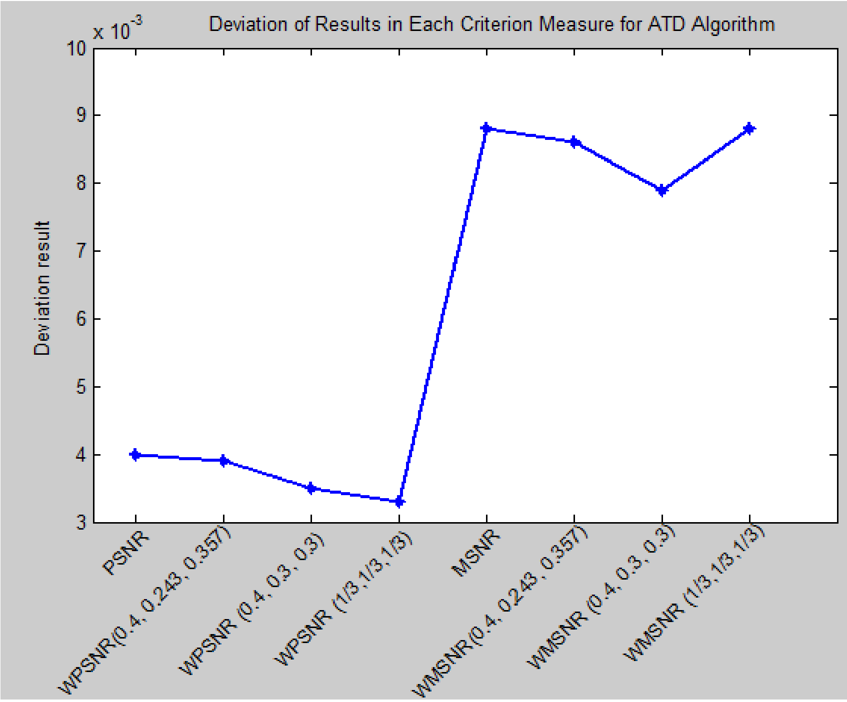


Figure 6.32: Deviation of Results for ATD

As we see from Figure 6.32 and Table 6.10, the best criterion measure to evaluation the quality of stego image is WPSNR with weight (1/3,1/3,1/3) it has the minimal value of deviation.

6.4 Performance of ATD and LSB for Color Images Depending on the Embedding Capacity

We studied the performance of the both algorithm (ATD and LSB) for color scale images by using WMSNR with weight (1/3, 1/3, 1/3) when changing the embedding capacity to 6 BPP and 10 BPP as shown in Figure 6.36.

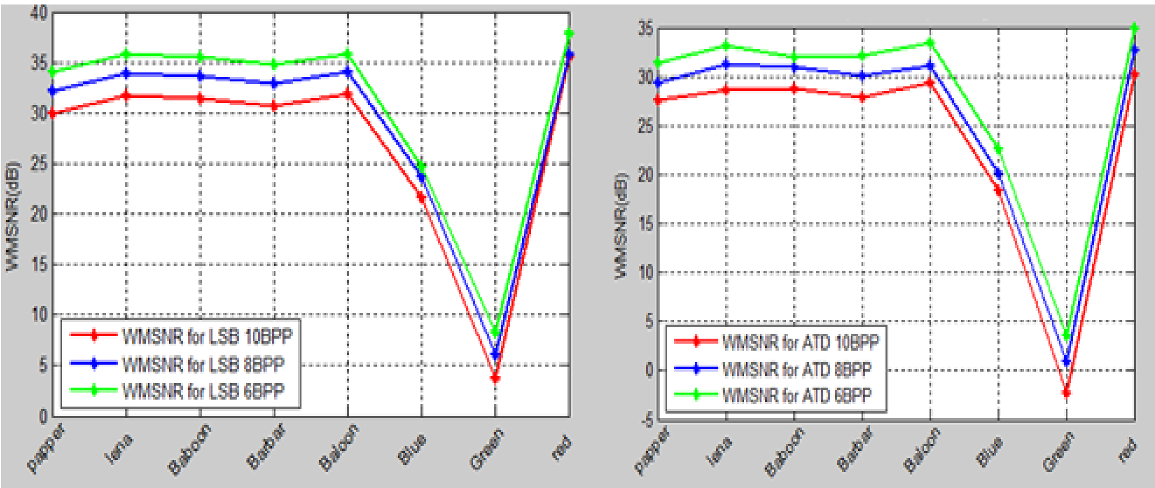


Figure 6.36: WMSNR (dB) Dependence on Embedding Capacity for LSB and ATD.

As we can see in Figure 5.36, WMSNR decreases when embedding capacity increases that complies with our expectations.

6.5 Summary of Chapter 6

Thus, in this chapter we have discussed and compared the gray scale images and color scale images results with criteria. We also compared these results with known experiments [28] presented in Section 2.5. Results obtained are as follows:

➢ For gray scale images:

1. We show PSNR for gray images for ATD and LSBCT, LSBDT.

2. We introduced such criteria as MSNR, APSNR, which have similar to PSNR curve shape.

➢ For color images:

1. We found that PSNR for color scale images behaves not as expected: for the same embedding capacity but for different combinations embedding into the same image we get different PSNR values, and different relations between the combinations.

2. We introduced such criteria as WMSNR, WPSNR which show stable behavior for different combinations. By evaluation of different metrics by LSB the minimal deviation we get in WMSNR with weight (1/3,1/3,1/3) the value of deviation is

0.211 and maximum result in WPSNR with weight (0.4, 0.243, 0.357) this weight we get from human color perception in Figure 2.15 as explain in Section 6.3.

3. We showed that stability of WMSNR is preserved when varying embedding capacity, and it decreases with the increase of embedding capacity.

Chapter 7

CONCLUSION AND FUTURE WORK

In this thesis, analysis, implementation, and experiments on steganographic methods (ATD, LSB, and LSBT) for the gray scale and color images are conducted. The algorithms are explained in details and analyzed. It was found out that LSBT method may work incorrectly (counter-examples are constructed). The reasons of the problems are understood, and LSBT is modified as LSBT-M imposing certain constraints on the LSBT parameters: threshold, and moduli values. LSBT-M algorithm is considered in static (LSBCT) and dynamic (LSBDT) variants. Dynamic variant is introduced to have experimental results compatible with those published in [28] as for LSBCT. The experiments are conducted on ATD, LSBCT, and LSBDT for gray scale images, and on ATD and LSB-based (LSB, ALSBmin, and ALSBmax) for color images Known quality measures of stego-images (PSNR, SNR) and proposed (MSNR, APSNR WMSNR, and WPSNR) are studied.

For gray scale images, the results are obtained for 15 gray scale cover images and random secret messages. According to our experiments, LSBDT is better than ATD in the quality of stego images, PSNR, when the embedding capacity EC is less than or equal to 3 BPP. In [28], LSBDT performance is shown as the performance of LSBT, but we show that LSBT (or, LSBCT after modification) has worse performance (compare Figures 6.6 and 2.17). Also, when MSNR and APSNR are applied for the both algorithms (LSBT and ATD) the results have the same form as that of PSNR.

For color images, the experiments are conducted with eight cover images with size 512*512 pixels, and one binary message as a secrete message. LSB and ATD are implemented for different embedding combinations of 8 bits embedding in each color pixel. According to experiments, PSNR of LSB for color images has fluctuations in different combinations with the same embedding capacity 8 bits in each pixel while the PSNR of ATD is stable in different combinations. When we apply WPSNR and MSNR and weight it by three different groups of weights, the result show that, the value of PSNR with weights (1/3, 1/3, 1/3) for LSB with different combinations is more plateaus than other weights. Also, the value of MSNR with weights (1/3, 1/3, 1/3) for LSB with different combinations looks invariant for different combinations when compared with other weights. MSNR with different weights for ATD is stable in different combinations as PSNR. For LSB algorithm when comparison between different metrics was made by deviation evaluation of the metrics, we got the best metric for LSB algorithm as WMSNR with weights (1/3,1/3,1/3) with the minimal deviation values as 0.211, and the maximum deviation was obtained for WPSNR with weights (0.4, 0.243, 0.357) that we got from human color perception. Thus, human color perception-originated weights are not appropriate for the images assessment, so we can conclude that human eyes and SNR-based metric presumably use different ways of image estimation.

Furthermore, when varying embedding capacity for LSB and ATD the results showed stability of WMSNR with weights (1/3, 1/3, 1/3), and it decreases with the increase of embedding capacity (we considered BPP=6, 8, and 10).

In the future work, we want to study LSBDT more in order to improve performance of LSBT for high embedding capacity so that it could beat ATD for all embedding capacity values.

.

REFERENCES

[1] Ankur, G., Gupta, S., & Bhushan, B. (2012). Information Hiding Using Least Significant Bit Steganography and Cryptography. *International Journal of Modern Education and Computer Science,* pp. 27-34. Vol. 5. No. 2.

[2] Agrawal, D ., & Samidha, D. (2013). Random Image Steganography in Spatial Domain. *IEEE, International Conference*. pp. 1-3. Vol. 12. No. 34.

[3] Acharya, U. D, Hemalatha, S, & Renuka, A. (2015). Wavelet Transform Based Steganography Technique to Hide Audio Signals in Image. *Procedia Computer Science,* pp. 272-281. Vol. 47. No. 4.

[4] Acharya, U. D, Hemalatha, S, Renuka, A & Kamath, P. R. (2013). A Secure Color Image Steganography in Transform Domain. *International Journal on Cryptography and Information Security (IJCIS)*, pp. 1304-3313. Vol. 3. No. 1.

[5] Cheng, S. C., & Wu, T. L. (2005). Sub Pixel Edge Detection of Color Images by Principal Axis Analysis and Moment-Preserving Principle. *Pattern Recognition,* pp. 527-537. Vol. 38. No. 4.

[6] Chang, C. C., Tai, W. L., & Lin, C. C. (2006). A Reversible Data Hiding Scheme Based on Side Match Vector Quantization. *IEEE Transactions on Circuits and Systems for Video Technology,* pp.1301-1308. Vol. 16. No.10.

[7] Chang, C. C., Chen, T. S., & Chung, L. Z. (2002). A Steganographic Method

Based upon JPEG and Quantization Table Modification. *Information Sciences*, pp. 123-138. Vol. 141. No. 1.

[8] Chen, B., Zhang, W., Ma, K., & Yu, N. (2014). Recursive Code Construction for Reversible Data Hiding in DCT Domain. *Multimedia Tools and Applications,* pp. 1985-2009. Vol. 72. No. 2.

[9] Chang, C. C., Hsiao, J. Y., & Chan, C. S. (2003). Finding Optimal Least-Significant-Bit Substitution in Image Hiding by Dynamic Programming Strategy. *Pattern Recognition*, pp. 1583-1595. Vol. 36. No.7.

[10] Chan, C. K., & Cheng, L. M. (2004). Hiding Data in Images by Simple LSB Substitution. *Pattern recognition*, pp. 469-474. Vol. 37. No.3.

[11] Cheddad, A., Condell, J., Curran, K., & Kevitt, P. M. (2010). Digital Image Steganography: Survey and Analysis of Current Methods. *Signal Processing*, pp. 727-752. Vol. 90. No.3.

[12] Chen, W. Y. (2007). Color Image Steganography Scheme Using Set Partitioning in Hierarchical Trees Coding, Digital Fourier Transform and Adaptive Phase Modulation. *Applied Mathematics and Computation*, pp. 432-448. Vol 185. No. 1.

[13] El-Kilani, W. S, Haweel, R. T., & Ramadan, H. H. (2014). A Fast Modified Signed Discrete Cosine Transform for Image Compression. *IEEE, International Conference on Computer Engineering & Systems (ICCES)*, pp. 56-61. Vol. 9.

No. 22.

[14] Gupta, N., & Sharma, N. (2013). Hiding Image in Audio Using DWT and LSB. *International Journal of Computer Applications*, pp. 11-14. Vol. 81. No. 2.

[15] Garg, S., & Mathur, M. (2014). Chaotic Map Based Steganography of Gray scale Images in Wavelet Domain. *IEEE, In Signal Processing and Integrated Networks (SPIN),* pp. 689-694. Vol. 10. No. 7.

[16] Hegde, R &Jagadeesha, S. (2015). Design and Implementation of Image Steganography by Using LSB Replacement Algorithm and Pseudo Random Encoding Technique. *International Journal on Recent and Innovation Trends in Computing and Communication,* pp. 4415 – 4420. Vol. 3. No. 7.

[17] Hussain, M., Wahab, A. W. A., Ho, A. T., Javed, N., & Jung, K. H. (2016). A Data Hiding Scheme Using Parity-Bit Pixel Value Differencing and Improved Rightmost Digit Replacement. *Signal Processing: Image Communication,* pp. 44-57. Vol. 50. No. 12.

[18] Jheng, Y. Z., Chen, C. Y., & Huang, C. F. (2015) Reversible Data Hiding Based on Histogram Modification over Ternary Computers.*Journal of Information Hiding and Multimedia Signal Processing,* pp .938-955. Vol. 6. No. 4.

[19] Kiruba, K., &Karthikeyan, S. (2013). Reliable Detection of Adaptive Pixel Pair Matching in Color and Grayscale Images. *IEEE, International Conferenceon Information Communication and Embedded Systems (ICICES)*. pp. 943-946.

Vol. 5. No. 9.

[20] Kinoshita, S. (2013).Bionanophotonics: an Introductory Textbook. CRC Press.

[21] Lavania, S., Matey, P. S., & Thanikaiselvan, V. (2014). Real-Time Implementation of Steganography in Medical Images Using Integer Wavelet Transform. *IEEE, International Conference on Computational Intelligence and Computing Research (ICCIC)*, pp. 1-5. Vol. 102. No. 77.

[22] Maji, A. K., Pal, R. K., & Roy, S. (2014). A Novel Steganographic Scheme Using Sudoku. *IEEE, International Conference on Electrical Information and Communication Technology (EICT)*, pp. 1-6. Vol. 14. No. 28.

[23] RGB color model. (2017, January 12).Retrieved from https://en.wikipedia.org/wiki/RGB_color_model.

[24] Singh, A., & Singh, H. (2015). An Improved LSB Based Image Steganography Technique for RGB Images. *IEEE, International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1-4. Vol. 16. No. 20.

[25] Taur, J. S., Lin, H. Y., Lee, H. L., & Tao, C. W. (2012). Data Hiding in DNA Sequences Based on Table Lookup Substitution. *International Journal of Innovative Computing, Information and Control*, pp. 6585-6598. Vol. 8. No. 10.

[26] Wang, S. J. (2005). Steganography of Capacity Required Using Modulo

Operator for Embedding Secret Image. *Applied Mathematics and Computation*, pp. 99-116. Vol. 164. No. 11.

[27] Wang, R. Z., Lin, C. F., & Lin, J. C. (2001). Image Hiding by Optimal LSB Substitution and Genetic Algorithm. *Pattern recognition*, pp. 671-683. Vol. 34. No. 3.

[28] Xu, W. L., Chang, C. C., Chen, T. S., & Wang, L. M. (2016). An Improved Least-Significant-Bit Substitution Method Using the Modulo Three Strategy. *Displays*, pp.36-42. Vol. 42. No. 17.

[29] Yu, Y. H., Chang, C. C., & Lin, I. C. (2007). A New Steganographic Method for Color and Grayscale Image Hiding. *Computer Vision and Image Understanding*, pp.183-194. Vol. 107. No. 3.

[30] Zhang, T., & Ping, X. (2003). A New Approach to Reliable Detection of LSB Steganography in Natural Images. *Signal Processing*, pp.2085-2093. Vol 83. No.10.

APPENDICES

Appendix A: ATD Algorithm for Gray Scale Images

Figure A.1. Cover Images Used in Gray Scale Images

A.1 The main program

```
% this program was written by Hajer A Al_aswed in 2016-2017 for ATD algorithm in gray scale [7] and its functions
clc
clear
cover_image=imread('C:\Users\hajer\Desktop\thesis\GRAY SCALE\ATD for text\home.gif');%Read cover image
[M N L]=size(cover_image);
S=sum(cover_image(:,:));
avg=sum(S)/(M*N);% calculate the mean of cover image
start_length =494288;   %start length of secreat message
disp('========================================================')
```

```
disp('cover_image  PSNR   Actual_PSNR  MSNR  MSE size_secret_data Bpp')
disp('            dB               dB              db ')
p='home.gif';
disp('=========================================================')
for h=1:11
increase_size=30000;
start_length=start_length+increase_size;
secret_massege= randi([0 1],1,start_length);%generate the secreat message
ternary_number=convertto__ternary(secret_massege);%convert the secreat message to ternary
[q size_secret_massege]=size(ternary_number);
stego_image=Embbedding(cover_image,size_secret_massege,ternary_number);%embedding function
MSE=0;
for i=1:M
for j=1:N
MSE=MSE+(double(cover_image(i,j))-double(stego_image(i,j)))^2; %calculate the MSE
end
end
MSE=MSE/(M*N);
PSNR=10*log10(255^2/(MSE));%calculate the PSNR
max_value=max(max(cover_image(:,:)));%calculate the maximum actual value
MSNR=10*log10(avg*avg/MSE);%calculate the MSNR
Bpp=(start_length)/(M*N);%calculate the BPP
Actual_PSNR=10*log10(double(max_value)*double(max_value)/MSE);%calculate the APSNR
disp(sprintf('%s %f %f %f %f %d%2f',p,PSNR, Actual_PSNR,MSNR ,MSE,start_length,Bpp));
end
disp('=========================================================')
%show the cover image and stego image
figure
subplot(2,2,[1,3]);
imshow(cover_image);
title('Cover image')
subplot(2,2,[2,4]);
imshow(stego_image);
title('Stego image')
output = extraction( stego_image,size_secret_massege );%extraction function
```

A.2 Convert the secret message to the ternary

```
function [ output ] = convertto__ternary( a )
[e r]=size(a);
k=1;
kk=1;
b_s=64;
w=mod(r,b_s);
if (w==0)
n=r/b_s;
else
n=(r-w)/b_s;
n=n+1;
end
for i=1:n
if (k+b_s-1)>=r
q=(k+b_s-1)-r;
for m=1:q
block(m)=0;
end
for m=q+1:b_s
block(m)=a(k);
k=k+1;
end
else
for m=1:b_s
block(m)=a(k);
k=k+1;
end
end
aa=convert_binary_decmal(block);
ternary=convert_decmal_ternary(aa);
[o p]=size(ternary);
for j=1:p
output(kk)=ternary(j);
kk=kk+1;
end
end
```

```
end
```

A.2.1 Convert the binary block to the decimal

```
function [ output ] = convert_binary_decmal( a )
[q k]=size(a);
k=k-1;
output=0;
for i=1:k+1
output=output+a(i)*power(2,k);
k=k-1;
end
end
```

A.2.2 Convert the decimal to the ternary

```
function [ output ] = convert_decmal_ternary( input )
kk=1;
k=1;
while (input~=0)
output1(kk)=mod(input,3);
input=fix(input/3);
kk=kk+1;
end
ss=41-(kk-1);
if (ss~=0)
for i=1:ss
output(i)=0;
end
for m=ss+1:41
output(m)=output1(k);
k=k+1;
end
else
for m=1:41
output(m)=output1(k);
k=k+1;
end
end
end
```

A.3 Embedding function

```
function [stego_image] = Embbedding( cover_image,size_secret_massege, ternary_number )
stego_image=cover_image;
k=1;
[M N]=size(cover_image);
for i=1:M
for j=1:N
cover_pixel_binary = dec2bin(cover_image(i,j),8);
for ii=1:6
sub1(ii)=cover_pixel_binary(ii);
end
sub1_dec=bin2dec(sub1);
sub2=cover_pixel_binary(7:8);
sub2_dec=bin2dec(sub2);
if (sub1_dec==63)
sub1_dec=62;
end
if (sub1_dec==0)
sub1_dec=1;
end
if (sub2_dec==3)
sub2_dec=2;
end
if (sub2_dec==0)
sub2_dec=1;
end
if(k<=size_secret_massege)
if (mod(sub1_dec,3)==ternary_number(k))
sub1_stego=sub1_dec;
elseif(mod(sub1_dec+1,3)==ternary_number(k))
sub1_stego=sub1_dec+1;
else
sub1_stego=sub1_dec-1;
end
end
k=k+1;
```

```
if(k<=size_secret_massege)
v=sub1_stego*4+sub2_dec;
if (mod(v,3)==ternary_number(k))
stego_image(i,j)=v;
elseif(mod(v+1,3)==ternary_number(k))
stego_image(i,j)=v+1;
else
stego_image(i,j)=v-1;
end
k=k+1;
end
end
end
end
```

A.4 Extraction function

```
function [ output ] = extraction( stego_image)
k=1;
q=1;
for i=1:512
for j=1:512
if (q<=size_secret_image)
cover_pixel_binary = dec2bin(stego_image(i,j),8);
sub1=cover_pixel_binary(1:6);
sub1_dec=bin2dec(sub1);
sub2=cover_pixel_binary(7:8);
sub2_dec=bin2dec(sub2);
output1(k)=mod(sub1_dec,3);
k=k+1;
output1(k)=mod(stego_image(i,j),3);
k=k+1;
q=q+1;
else
continue
end
end
end
```

```
[e r]=size(output1);
k=1;
b_s=41;
w=mod(r,b_s);
if (w==0)
n=r/b_s;
else
n=(r-w)/b_s;
n=n+1;
end
for i=1:n
if (k+b_s-1)>=r
q=(k+b_s-1)-r;
for m=1:q
block(m)=0;
end
for m=q+1:b_s
block(m)=output1(k);
k=k+1;
end
else
for m=1:b_s
block(m)=output1(k);
k=k+1;
end
end
aa=conver_ternary_decimal(block);
output=convert_decmal_binary(aa);
end
end
```

A.4.1 Convert_ternary to decimal function

```
function [ output] = conver_ternary_decimal(a)
[q k]=size(a);
k=k-1;
output=0;
for i=1:k+1
```

```
output=output+a(i)*power(3,k);
k=k-1;
end
end
```

A.4.2 Convert decimal to binary function

```
function [ output ] = convert_decmal_binary( input )
kk=1;
k=1;
while (input~=0)
output1(kk)=mod(input,2);
input=fix(input/2);
kk=kk+1;
end
ss=64-(kk-1);
if (ss~=0)
for i=1:ss
output(i)=0;
end
for m=ss+1:64
output(m)=output1(k);
k=k+1;
end
else
for m=1:64
output(m)=output1(k);
k=k+1;
end
end
end
```

A.5 Draw graph program

```
clc
clear
pos=[2 2.2 2.4 2.6 2.8 3 3.2];
PSNR_ATD=[39.32 38.86 38.44 38.04 37.87 37.69 37.41];
PSNR_T=[42.18 41.75 36.64 36.28 36.11 35.95 35.65];
```

```
PSNR_adapter_T=[43.21 41.76 40.04 39.03 38.55 38.05 35.60];
MSNR_ATD=[33.06 32.60 32.18 31.78 31.62 31.28 31.15];
MSNR_T=[35.93 35.49 30.38 30.02 29.85 29.69 29.39];
MSNR_adapter_T=[36.95 35.50 33.78 32.77 32.29 31.80 29.34];
A_PSNR_ATD=[39.34 38.86 38.44 38.06 37.87 37.70 37.40];
A_PSNR_T=[41.86 41.42 36.32 35.93 35.75 35.60 35.30];
A_PSNR_adapter_T=[42.85 41.43 39.69 38.69 38.21 37.70 35.28];
figure (1)
h1 = plot(pos,PSNR_ATD,'g','LineWidth',2,'Marker','*');
hold on
h2 = plot(pos,PSNR_T,'b','LineWidth',2,'Marker','*');
hold on
h3 = plot(pos,PSNR_adapter_T,'k','LineWidth',2,'Marker','*');
hold on
legend([h1,h2,h3],'PSNR ATD','PSNR T=160','PSNR adapter T','Location' ,'southwest');
axis([1.95 3.3  30 45])
title('PSNR for Lean image')
ylabel('PSNR(dB)');
xlabel('BPP');
grid
figure (2)
h3 = plot(pos,MSNR_ATD,'g','LineWidth',2,'Marker','*');
hold on
h4 = plot(pos,MSNR_T,'b','LineWidth',2,'Marker','*');
hold on
h5 = plot(pos,MSNR_adapter_T,'k','LineWidth',2,'Marker','*');
hold on
legend([h3,h4,h5],'MSNR ATD','MSNR T=160','MSNR adapter T','Location' ,'southwest');
title('MSNR for Lean image')
ylabel('MSNR(dB)');
xlabel('BPP');
axis([1.95 3.3  27 40])
grid on;
figure (3)
h1 = plot(pos,A_PSNR_ATD,'g','LineWidth',2,'Marker','*');
hold on
```

```
h2 = plot(pos,A_PSNR_T,'b','LineWidth',2,'Marker','*');
hold on
h3 = plot(pos,A_PSNR_adapter_T,'k','LineWidth',2,'Marker','*');
hold on
legend([h1,h2,h3],'APSNR ATD','APSNR T=160','APSNR adapter T', 'Location','southwest');
axis([1.95 3.3  30 45])
title('Actual PSNR for Lean image')
ylabel('APSNR(dB)');
xlabel('BPP');
grid on
```

A.6 Screenshots of ATD for gray scale images results for different cover images with size 512×512

Results for Lena image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
lena.BMP	39.341881	38.994399	33.082488	7.566452	524288	2.000000
lena.BMP	39.093979	38.746497	32.834586	8.010921	554288	2.114441
lena.BMP	38.863923	38.516441	32.604530	8.446720	584288	2.228882
lena.BMP	38.662911	38.315429	32.403518	8.846863	614288	2.343323
lena.BMP	38.435312	38.087830	32.175919	9.322861	644288	2.457764
lena.BMP	38.245145	37.897663	31.985753	9.740154	674288	2.572205
lena.BMP	38.055621	37.708139	31.796228	10.174622	704288	2.686646
lena.BMP	37.874323	37.526841	31.614930	10.608356	734288	2.801086
lena.BMP	37.695110	37.347628	31.435717	11.055271	764288	2.915527
lena.BMP	37.531930	37.184449	31.272538	11.478558	794288	3.029968
lena.BMP	37.396861	37.049379	31.137469	11.841160	824288	3.144409

Results for Zelda image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
zelda.BMP	39.337935	38.403841	32.097491	7.573330	524288	2.000000
zelda.BMP	39.099848	38.165754	31.859404	8.000103	554288	2.114441
zelda.BMP	38.864833	37.930739	31.624389	8.444950	584288	2.228882
zelda.BMP	38.628794	37.694700	31.388351	8.916634	614288	2.343323
zelda.BMP	38.431969	37.497875	31.191525	9.330040	644288	2.457764
zelda.BMP	38.242105	37.308011	31.001661	9.746975	674288	2.572205
zelda.BMP	38.060954	37.126860	30.820510	10.162136	704288	2.686646
zelda.BMP	37.853653	36.919559	30.613209	10.658966	734288	2.801086
zelda.BMP	37.697112	36.763018	30.456668	11.050175	764288	2.915527
zelda.BMP	37.518664	36.584570	30.278220	11.513676	794288	3.029968
zelda.BMP	37.396839	36.462745	30.156395	11.841221	824288	3.144409

Results for Airplane image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
airplane.BMP	39.329231	38.470667	36.264165	7.588524	524288	2.000000
airplane.BMP	39.098864	38.240300	36.033798	8.001915	554288	2.114441
airplane.BMP	38.871755	38.013191	35.806690	8.431499	584288	2.228882
airplane.BMP	38.654412	37.795848	35.589346	8.864193	614288	2.343323
airplane.BMP	38.448387	37.589823	35.383321	9.294834	644288	2.457764
airplane.BMP	38.238610	37.380046	35.173544	9.754822	674288	2.572205
airplane.BMP	38.054823	37.196259	34.989757	10.176491	704288	2.686646
airplane.BMP	37.852156	36.993592	34.787090	10.662640	734288	2.801086
airplane.BMP	37.686961	36.828397	34.621895	11.076035	764288	2.915527
airplane.BMP	37.528986	36.670422	34.463920	11.486343	794288	3.029968
airplane.BMP	37.393042	36.534478	34.327976	11.851578	824288	3.144409

Results for Baboon image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
Baboon.BMP	39.326553	38.727288	33.417389	7.593204	524288	2.000000
Baboon.BMP	39.072292	38.473027	33.163128	8.051025	554288	2.114441
Baboon.BMP	38.865209	38.265945	32.956046	8.444218	584288	2.228882
Baboon.BMP	38.644590	38.045326	32.735426	8.884262	614288	2.343323
Baboon.BMP	38.449114	37.849850	32.539951	9.293278	644288	2.457764
Baboon.BMP	38.251733	37.652468	32.342569	9.725391	674288	2.572205
Baboon.BMP	38.039026	37.439762	32.129863	10.213573	704288	2.686646
Baboon.BMP	37.871320	37.272056	31.962157	10.615692	734288	2.801086
Baboon.BMP	37.682522	37.083258	31.773358	11.087360	764288	2.915527
Baboon.BMP	37.535643	36.936378	31.626479	11.468750	794288	3.029968
Baboon.BMP	37.400327	36.801062	31.491163	11.831715	824288	3.144409

Results for Barbara image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
barbara.bmp	39.333619	39.161615	33.250531	7.580860	524288	2.000000
barbara.bmp	39.095762	38.923758	33.012674	8.007633	554288	2.114441
barbara.bmp	38.868607	38.696603	32.785519	8.437614	584288	2.228882
barbara.bmp	38.648457	38.476454	32.565370	8.876354	614288	2.343323
barbara.bmp	38.435383	38.263379	32.352295	9.322708	644288	2.457764
barbara.bmp	38.230151	38.058147	32.147063	9.773842	674288	2.572205
barbara.bmp	38.051762	37.879759	31.968674	10.183666	704288	2.686646
barbara.bmp	37.865821	37.693818	31.782734	10.629143	734288	2.801086
barbara.bmp	37.693234	37.521230	31.610146	11.060047	764288	2.915527
barbara.bmp	37.533033	37.361030	31.449946	11.475643	794288	3.029968
barbara.bmp	37.422892	37.250889	31.339805	11.770397	824288	3.144409

Results for Elain image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
elain.BMP	39.341942	39.135125	33.904541	7.566345	524288	2.000000
elain.BMP	39.087558	38.880741	33.650157	8.022774	554288	2.114441
elain.BMP	38.870292	38.663475	33.432890	8.434341	584288	2.228882
elain.BMP	38.637765	38.430948	33.200363	8.898235	614288	2.343323
elain.BMP	38.427322	38.220506	32.989921	9.340027	644288	2.457764
elain.BMP	38.250305	38.043489	32.812904	9.728588	674288	2.572205
elain.BMP	38.053759	37.846942	32.616357	10.178986	704288	2.686646
elain.BMP	37.882564	37.675747	32.445162	10.588245	734288	2.801086
elain.BMP	37.703250	37.496433	32.265848	11.034569	764288	2.915527
elain.BMP	37.523998	37.317181	32.086597	11.499542	794288	3.029968
elain.BMP	37.384919	37.178103	31.947518	11.873764	824288	3.144409

Results for Goldhill image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
goldhill.BMP	39.342923	38.670360	32.181755	7.564636	524288	2.000000
goldhill.BMP	39.100264	38.427701	31.939095	7.999336	554288	2.114441
goldhill.BMP	38.858217	38.185653	31.697048	8.457825	584288	2.228882
goldhill.BMP	38.644797	37.972233	31.483628	8.883839	614288	2.343323
goldhill.BMP	38.443864	37.771300	31.282695	9.304520	644288	2.457764
goldhill.BMP	38.238009	37.565445	31.076840	9.756172	674288	2.572205
goldhill.BMP	38.038970	37.366406	30.877801	10.213707	704288	2.686646
goldhill.BMP	37.876530	37.203966	30.715361	10.602966	734288	2.801086
goldhill.BMP	37.689127	37.016563	30.527958	11.070511	764288	2.915527
goldhill.BMP	37.533489	36.860926	30.372321	11.474438	794288	3.029968
goldhill.BMP	37.391731	36.719167	30.230562	11.855156	824288	3.144409

Results for Peppers image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
peppers.BMP	39.318747	38.609300	32.793501	7.606865	524288	2.000000
peppers.BMP	39.089925	38.380479	32.564680	8.018402	554288	2.114441
peppers.BMP	38.861487	38.152041	32.336242	8.451458	584288	2.228882
peppers.BMP	38.639752	37.930305	32.114506	8.894165	614288	2.343323
peppers.BMP	38.426159	37.716713	31.900913	9.342529	644288	2.457764
peppers.BMP	38.235772	37.526325	31.710526	9.761200	674288	2.572205
peppers.BMP	38.043687	37.334241	31.518442	10.202618	704288	2.686646
peppers.BMP	37.855876	37.146430	31.330631	10.653511	734288	2.801086
peppers.BMP	37.688403	36.978956	31.163157	11.072357	764288	2.915527
peppers.BMP	37.530633	36.821187	31.005388	11.481987	794288	3.029968
peppers.BMP	37.403005	36.693558	30.877759	11.824421	824288	3.144409

Results for Lady image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
lady.tif	39.292160	38.656323	32.618740	7.653576	524288	2.000000
lady.tif	39.072905	38.437068	32.399485	8.049889	554288	2.114441
lady.tif	38.839293	38.203456	32.165874	8.494759	584288	2.228882
lady.tif	38.620642	37.984805	31.947222	8.933388	614288	2.343323
lady.tif	38.404545	37.768708	31.731125	9.389141	644288	2.457764
lady.tif	38.209611	37.573775	31.536192	9.820175	674288	2.572205
lady.tif	38.012706	37.376869	31.339286	10.275661	704288	2.686646
lady.tif	37.832198	37.196361	31.158778	10.711754	734288	2.801086
lady.tif	37.665889	37.030052	30.992469	11.129906	764288	2.915527
lady.tif	37.494818	36.858981	30.821398	11.577068	794288	3.029968
lady.tif	37.368823	36.732986	30.695403	11.917854	824288	3.144409

Results for House image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
house.tif	39.303155	39.234761	32.832395	7.634224	524288	2.000000
house.tif	39.094649	39.026256	32.623890	8.009686	554288	2.114441
house.tif	38.856875	38.788482	32.386116	8.460438	584288	2.228882
house.tif	38.631489	38.563096	32.160730	8.911102	614288	2.343323
house.tif	38.414849	38.346456	31.944090	9.366890	644288	2.457764
house.tif	38.218322	38.149929	31.747563	9.800499	674288	2.572205
house.tif	38.017209	37.948816	31.546450	10.265011	704288	2.686646
house.tif	37.849750	37.781357	31.378991	10.668549	734288	2.801086
house.tif	37.687390	37.618997	31.216631	11.074940	764288	2.915527
house.tif	37.514217	37.445824	31.043458	11.525471	794288	3.029968
house.tif	37.390053	37.321660	30.919294	11.859737	824288	3.144409

Results for Home image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
home.gif	39.252201	39.149409	27.894122	7.724319	524288	2.000000
home.gif	38.940786	38.837993	27.582707	8.298542	554288	2.114441
home.gif	38.701533	38.598740	27.343454	8.768536	584288	2.228882
home.gif	38.469623	38.366830	27.111544	9.249496	614288	2.343323
home.gif	38.269625	38.166833	26.911547	9.685406	644288	2.457764
home.gif	38.047403	37.944610	26.689324	10.193893	674288	2.572205
home.gif	37.811679	37.708887	26.453600	10.762482	704288	2.686646
home.gif	37.591511	37.488718	26.233432	11.322159	734288	2.801086
home.gif	37.408363	37.305570	26.050284	11.809841	764288	2.915527
home.gif	37.261233	37.158440	25.903154	12.216789	794288	3.029968
home.gif	37.119336	37.016543	25.761257	12.622540	824288	3.144409

Results for Camerman image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
camerman.tif	39.330827	39.330827	32.660746	7.585735	524288	2.000000
camerman.tif	39.085099	39.085099	32.415019	8.027317	554288	2.114441
camerman.tif	38.854452	38.854452	32.184371	8.465160	584288	2.228882
camerman.tif	38.631104	38.631104	31.961024	8.911892	614288	2.343323
camerman.tif	38.444119	38.444119	31.774038	9.303974	644288	2.457764
camerman.tif	38.247682	38.247682	31.577601	9.734467	674288	2.572205
camerman.tif	38.046189	38.046189	31.376108	10.196743	704288	2.686646
camerman.tif	37.863604	37.863604	31.193523	10.634571	734288	2.801086
camerman.tif	37.700131	37.700131	31.030051	11.042496	764288	2.915527
camerman.tif	37.514888	37.514888	30.844808	11.523689	794288	3.029968
camerman.tif	37.405411	37.405411	30.735330	11.817871	824288	3.144409

Results for Boy image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
BOY.tif	39.325992	39.325992	31.506344	7.594185	524288	2.000000
BOY.tif	39.079071	39.079071	31.259423	8.038467	554288	2.114441
BOY.tif	38.869261	38.869261	31.049613	8.436344	584288	2.228882
BOY.tif	38.643311	38.643311	30.823663	8.886879	614288	2.343323
BOY.tif	38.424377	38.424377	30.604729	9.346363	644288	2.457764
BOY.tif	38.235546	38.235546	30.415898	9.761707	674288	2.572205
BOY.tif	38.049001	38.049001	30.229353	10.190144	704288	2.686646
BOY.tif	37.858396	37.858396	30.038748	10.647331	734288	2.801086
BOY.tif	37.681897	37.681897	29.862250	11.088955	764288	2.915527
BOY.tif	37.509734	37.509734	29.690087	11.537373	794288	3.029968
BOY.tif	37.389279	37.389279	29.569631	11.861851	824288	3.144409

>> |

Results for Boat image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
boat.gif	39.332039	38.769193	33.880109	7.583618	524288	2.000000
boat.gif	39.088399	38.525553	33.636468	8.021221	554288	2.114441
boat.gif	38.856862	38.294016	33.404931	8.460464	584288	2.228882
boat.gif	38.645383	38.082537	33.193452	8.882641	614288	2.343323
boat.gif	38.438958	37.876112	32.987027	9.315037	644288	2.457764
boat.gif	38.235873	37.673028	32.783943	9.760971	674288	2.572205
boat.gif	38.043126	37.480280	32.591195	10.203938	704288	2.686646
boat.gif	37.875824	37.312978	32.423893	10.604691	734288	2.801086
boat.gif	37.688584	37.125738	32.236653	11.071896	764288	2.915527
boat.gif	37.522932	36.960087	32.071002	11.502365	794288	3.029968
boat.gif	37.397419	36.834574	31.945489	11.839638	824288	3.144409

Results for Baby image in Figure A.1

cover_image	PSNR dB	Actual_PSNR dB	MSNR db	MSE	size_secret_data	Bpp
BABY.png	39.251485	39.148692	34.773021	7.725594	524288	2.000000
BABY.png	39.016886	38.914093	34.538422	8.154396	554288	2.114441
BABY.png	38.786774	38.683981	34.308310	8.598110	584288	2.228882
BABY.png	38.580066	38.477273	34.101602	9.017242	614288	2.343323
BABY.png	38.367881	38.265088	33.889417	9.468742	644288	2.457764
BABY.png	38.180176	38.077383	33.701712	9.886959	674288	2.572205
BABY.png	37.989785	37.886992	33.511321	10.330036	704288	2.686646
BABY.png	37.818354	37.715562	33.339890	10.745953	734288	2.801086
BABY.png	37.634706	37.531914	33.156242	11.210106	764288	2.915527
BABY.png	37.463726	37.360933	32.985262	11.660248	794288	3.029968
BABY.png	37.345623	37.242830	32.867159	11.981689	824288	3.144409

Appendix B: LSB with Constant Threshold Algorithm for Gray Scale (LSBCT)

B.1 The main program

```
% this program was written by Hajer A Al_aswed in 2016-2017
for LSB with Constant threshold algorithm in gray scale [8] and its functions

clc
clear
cover_image=imread('C:\Users\hajer\Desktop\thesis\GRAY SCALE\MA for text\home.gif');%Read cover image
[M N L]=size(cover_image);
S=sum(cover_image(:,:));
avg=sum(S)/(M*N);% calculate the mean of cover image
disp('==========================================================')
disp('cover_image  Size_secret_data PSNR  Actual_PSNR   MSNR MSE Ml Mu T ')
disp('          dB              dB                  db ')
p='home.gif';
disp('==========================================================')
ml=4;
mu=8;
T=160;
start_length =494288;   %start length of secreat message
increase_size=30000;
for q=1:2
start_length=start_length+increase_size;
secret_massege= randi([0 1],1,start_length);%generate the secreat message
[e r]=size(secret_massege);
stego_image=Embedding( ml,mu,T,cover_image,secret_massege); %embedding function
MSE=0;
for i=1:M
for j=1:N
MSE=MSE+(double(cover_image(i,j))-double(stego_image(i,j)))^2; %calculate the MSE
end
end
MSE=MSE/(M*N);
PSNR=10*log10(255^2/(MSE));%calculate the PSNR
max_value=max(max(cover_image(:,:)));%calculate the maximum actual value
```

```
MSNR=10*log10(avg*avg/MSE);%calculate the MSNR
Bpp=(start_length)/(M*N);%calculate the BPP
Actual_PSNR=10*log10(double(max_value)*double(max_value)/MSE);%calculate the APSNR
disp(sprintf('%s %d  %f  %f %f  %f%d  %d %d' ,p,start_length,PSNR,
Actual_PSNR,MSNR,MSE,ml,mu,T));
end
ml=8;
mu=16;
for q=1:5
start_length=start_length+increase_size;
secret_massege= randi([0 1],1,start_length);%generate the secreat message
[e r]=size(secret_massege);
stego_image=Embedding( ml,mu,T,cover_image,secret_massege); %embedding function
MSE=0;
for i=1:M
for j=1:N
MSE=MSE+(double(cover_image(i,j))-double(stego_image(i,j)))^2; %calculate the MSE
end
end
MSE=MSE/(M*N);
PSNR=10*log10(255^2/(MSE));%calculate the PSNR
max_value=max(max(cover_image(:,:)));%calculate the maximum actual value
MSNR=10*log10(avg*avg/MSE);%calculate the MSNR
Bpp=(start_length)/(M*N);%calculate the BPP
Actual_PSNR=10*log10(double(max_value)*double(max_value)/MSE);%calculate the APSNR
disp(sprintf('%s %d  %f  %f %f  %f%d  %d %d' ,p,start_length,PSNR,
Actual_PSNR,MSNR,MSE,ml,mu,T));
end
disp('=========================================================')
%show the cover image and stego image
figure
subplot(2,2,[1,3]);
imshow(cover_image);
title('Cover image')
subplot(2,2,[2,4]);
imshow(stego_image);
title('Stego image')
```

```
output = extraction( stego_image,T,mu,ml);%extraction function
```

B.2 Embedding function

```
function [ stego_image ] = Embedding( ml,mu,T,cover_image,secret_massege)
[e r]=size(secret_massege);
[M N L]=size(cover_image);
stego_image=cover_image;
m=0;
mm=0;
k=1;
for i=1:M
for j=1:N
if (k<=r)
if (cover_image(i,j)>=T)
m=m+1;
EC=log2(mu);  %how many bit will be embedding in the pixel
RES=mod(cover_image(i,j),mu);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
D=abs(RES-DEC);
if (cover_image(i,j)>(255-(mu/2)+1))
stego_image(i,j)=(255-mu+1)+DEC;
end
if((T+(mu/2))<cover_image(i,j)<=(255-(mu/2)+1))
if (D>(mu/2))
AV=mu-D;
if (RES>DEC)
stego_image(i,j)=cover_image(i,j)+AV;
else
```

```
stego_image(i,j)=cover_image(i,j)-AV;
end
end
if(D<=(mu/2))
AV=D;
if (RES>DEC)
stego_image(i,j)=cover_image(i,j)-AV;
else
stego_image(i,j)=cover_image(i,j)+AV;
end
end
end
if(T<=cover_image(i,j)<=(T+(mu/2)))
stego_image(i,j)=cover_image(i,j)-RES+DEC;
end
end
if (cover_image(i,j)<T)
mm=mm+1;
EC=log2(ml);
RES=mod(cover_image(i,j),ml);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
D=abs(RES-DEC);
if (cover_image(i,j)<(ml/2))
stego_image(i,j)=DEC;
end
if((ml/2)<=cover_image(i,j)<(T-(ml/2)))
if (D>(ml/2))
```

```
AV=ml-D;
if (RES>DEC)
stego_image(i,j)=cover_image(i,j)+AV;
else
stego_image(i,j)=cover_image(i,j)-AV;
end
end
if(D<=(ml/2))
AV=D;
if (RES>DEC)
stego_image(i,j)=cover_image(i,j)-AV;
else
stego_image(i,j)=cover_image(i,j)+AV;
end
end
end
if((T-(ml/2))<=cover_image(i,j)<T)
stego_image(i,j)=cover_image(i,j)-RES+DEC;
end
end
end
end
end
end
```

B.3 Extraction function

```
function [ output] = extraction( stego_image,T,mu,ml)
[M N]=size(stego_image);
kk=1;
k=1;
for i=1:M
for j=1:N
if (kk<=r)
if (stego_image(i,j)>=T)
EC=log2(mu);
RES=mod(stego_image(i,j),mu);
if (k+EC-1)>=r
```

```
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
else
EC=log2(ml);
RES=mod(stego_image(i,j),ml);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
```

```
end
end
end
end
end
```

B.4 Screenshots of LSBCT Algorithm, T=160, for gray scale results for different cover images with size 512×512

Results for Lena image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	M1	Mu	T
lenaBMP.bnp	524288	42.203832	41.856350	35.944439	3.914688	4	8	160
lenaBMP.bnp	584288	41.771417	41.423935	35.512024	4.324528	4	8	160
lenaBMP.bnp	644288	36.664771	36.317289	30.405378	14.015327	8	16	160
lenaBMP.bnp	704288	36.279082	35.931600	30.019690	15.316944	8	16	160
lenaBMP.bnp	734288	36.106057	35.758575	29.846664	15.939499	8	16	160
lenaBMP.bnp	764288	35.947297	35.599815	29.687904	16.532963	8	16	160
lenaBMP.bnp	824288	35.648520	35.301038	29.389127	17.710400	8	16	160

Results for Zelda image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	M1	Mu	T
zelda.bmp	524288	42.939214	42.005120	35.698770	3.304905	4	8	160
zelda.bmp	584288	42.681026	41.746932	35.440582	3.507339	4	8	160
zelda.bmp	644288	37.428747	36.494653	30.188304	11.754539	8	16	160
zelda.bmp	704288	37.096911	36.162817	29.856467	12.687885	8	16	160
zelda.bmp	734288	36.964019	36.029925	29.723575	13.082130	8	16	160
zelda.bmp	764288	36.748948	35.814854	29.508504	13.746292	8	16	160
zelda.bmp	824288	36.441192	35.507098	29.200748	14.755745	8	16	160

Results for Airplane image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	M1	Mu	T
airplane.bmp	524288	40.217775	39.359211	37.152709	6.184475	4	8	160
airplane.bmp	584288	39.785172	38.926608	36.720106	6.832241	4	8	160
airplane.bmp	644288	34.543641	33.685077	31.478575	22.841061	8	16	160
airplane.bmp	704288	34.220821	33.362257	31.155755	24.603577	8	16	160
airplane.bmp	734288	34.060481	33.201917	30.995415	25.528912	8	16	160
airplane.bmp	764288	33.912036	33.053472	30.846970	26.416592	8	16	160
airplane.bmp	824288	33.587272	32.728707	30.522206	28.467754	8	16	160

Results for Baboon image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	M1	Mu	T
Baboon.bmp	524288	41.955945	41.356681	36.046782	4.144630	4	8	160
Baboon.bmp	584288	41.486587	40.887323	35.577423	4.617657	4	8	160
Baboon.bmp	644288	36.553475	35.954210	30.644311	14.379139	8	16	160
Baboon.bmp	704288	36.170410	35.571146	30.261246	15.705051	8	16	160
Baboon.bmp	734288	35.985098	35.385834	30.075934	16.389683	8	16	160
Baboon.bmp	764288	35.784355	35.185090	29.875191	17.165043	8	16	160
Baboon.bmp	824288	35.431005	34.831741	29.521841	18.620007	8	16	160

Results for Barbara image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
barbara.png	524288	41.268970	41.096966	35.185882	4.854935	4	8	160
barbara.png	584288	40.890622	40.718619	34.807534	5.296856	4	8	160
barbara.png	644288	35.606039	35.434035	29.522951	17.884487	8	16	160
barbara.png	704288	35.307640	35.135636	29.224552	19.156509	8	16	160
barbara.png	734288	35.190916	35.018913	29.107829	19.678352	8	16	160
barbara.png	764288	35.067682	34.895678	28.984594	20.244740	8	16	160
barbara.png	824288	34.822927	34.650924	28.739840	21.418430	8	16	160

Results for Elain image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
elain.bmp	524288	41.833421	41.626604	36.396019	4.263226	4	8	160
elain.bmp	584288	41.289746	41.082930	35.852345	4.831764	4	8	160
elain.bmp	644288	36.548855	36.342038	31.111453	14.394444	8	16	160
elain.bmp	704288	36.121415	35.914598	30.684013	15.883232	8	16	160
elain.bmp	734288	35.904040	35.697224	30.466639	16.698456	8	16	160
elain.bmp	764288	35.669586	35.462769	30.232184	17.624702	8	16	160
elain.bmp	824288	35.289679	35.082862	29.852278	19.235897	8	16	160

Results for Goldhill image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
goldhill.bmp	524288	42.626505	41.953942	35.465337	3.551647	4	8	160
goldhill.bmp	584288	42.368497	41.695934	35.207328	3.769039	4	8	160
goldhill.bmp	644288	37.180639	36.508076	30.019470	12.445618	8	16	160
goldhill.bmp	704288	36.863338	36.190775	29.702170	13.388950	8	16	160
goldhill.bmp	734288	36.717844	36.045280	29.556675	13.845097	8	16	160
goldhill.bmp	764288	36.610808	35.938245	29.449640	14.190559	8	16	160
goldhill.bmp	824288	36.363859	35.691296	29.202691	15.020847	8	16	160

Results for Peppers image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
peppers.bmp	524288	41.931976	41.222530	35.406731	4.167568	4	8	160
peppers.bmp	584288	41.409715	40.700269	34.884470	4.700119	4	8	160
peppers.bmp	644288	36.395454	35.686008	29.870208	14.911968	8	16	160
peppers.bmp	704288	36.065148	35.355702	29.539903	16.090351	8	16	160
peppers.bmp	734288	35.902141	35.192695	29.376896	16.705761	8	16	160
peppers.bmp	764288	35.760191	35.050744	29.234945	17.260815	8	16	160
peppers.bmp	824288	35.348516	34.639069	28.823270	18.977055	8	16	160

Results for Lady image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
lady.tif	524288	41.160990	40.525153	34.487570	4.977158	4	8	160
lady.tif	584288	40.771567	40.135731	34.098148	5.444069	4	8	160
lady.tif	644288	35.443139	34.807302	28.769719	18.568058	8	16	160
lady.tif	704288	35.136196	34.500359	28.462777	19.927864	8	16	160
lady.tif	734288	34.953111	34.317275	28.279692	20.785919	8	16	160
lady.tif	764288	34.798643	34.162807	28.125224	21.538528	8	16	160
lady.tif	824288	34.598726	33.962889	27.925306	22.553181	8	16	160

Results for House image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
house.tif	524288	42.789644	42.721251	36.318885	3.420708	4	8	160
house.tif	584288	42.363169	42.294775	35.892409	3.773666	4	8	160
house.tif	644288	37.169895	37.101502	30.699136	12.476444	8	16	160
house.tif	704288	36.889674	36.821280	30.418914	13.308006	8	16	160
house.tif	734288	36.742684	36.674291	30.271925	13.766132	8	16	160
house.tif	764288	36.613554	36.545161	30.142795	14.181591	8	16	160
house.tif	824288	36.217127	36.148734	29.746368	15.537018	8	16	160

Results for Home image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
home.gif	524288	43.779233	43.676440	32.421154	2.723686	4	8	160
home.gif	584288	43.739767	43.636974	32.381688	2.748550	4	8	160
home.gif	644288	38.723742	38.620950	27.365664	8.723808	8	16	160
home.gif	704288	38.308204	38.205411	26.950125	9.599751	8	16	160
home.gif	734288	38.123626	38.020833	26.765547	10.016541	8	16	160
home.gif	764288	37.919640	37.816847	26.561561	10.498238	8	16	160
home.gif	824288	37.780066	37.677273	26.421987	10.841110	8	16	160

Results for Camerman image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
camerman.tif	524288	41.300901	41.300901	34.630820	4.819370	4	8	160
camerman.tif	584288	41.025578	41.025578	34.355497	5.134789	4	8	160
camerman.tif	644288	35.626847	35.626847	28.956766	17.799004	8	16	160
camerman.tif	704288	35.409521	35.409521	28.739440	18.712349	8	16	160
camerman.tif	734288	35.350584	35.350584	28.680504	18.968018	8	16	160
camerman.tif	764288	35.223804	35.223804	28.553724	19.529896	8	16	160
camerman.tif	824288	35.036920	35.036920	28.366839	20.388645	8	16	160

Results for Boy image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
BOY.tif	524288	42.552515	42.552515	34.732867	3.612675	4	8	160
BOY.tif	584288	42.110142	42.110142	34.290494	4.000057	4	8	160
BOY.tif	644288	37.309155	37.309155	29.489507	12.082726	8	16	160
BOY.tif	704288	36.808396	36.808396	28.988748	13.559410	8	16	160
BOY.tif	734288	36.613498	36.613498	28.793850	14.181774	8	16	160
BOY.tif	764288	36.386613	36.386613	28.566965	14.942356	8	16	160
BOY.tif	824288	36.025221	36.025221	28.205574	16.238960	8	16	160

Results for Boat image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
boat.gif	524288	40.747378	40.184533	35.295448	5.474476	4	8	160
boat.gif	584288	40.488976	39.926130	35.037045	5.810089	4	8	160
boat.gif	644288	34.937442	34.374596	29.485512	20.861050	8	16	160
boat.gif	704288	34.672112	34.109267	29.220182	22.175282	8	16	160
boat.gif	734288	34.578281	34.015435	29.126350	22.659603	8	16	160
boat.gif	764288	34.513507	33.950662	29.061577	23.000095	8	16	160
boat.gif	824288	34.310001	33.747155	28.858071	24.103508	8	16	160

Results for Baby image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
BABY.png	524288	40.818714	40.715921	36.340250	5.385288	4	8	160
BABY.png	584288	40.410863	40.308071	35.932399	5.915535	4	8	160
BABY.png	644288	35.109098	35.006305	30.630634	20.052593	8	16	160
BABY.png	704288	34.793511	34.690718	30.315047	21.563999	8	16	160
BABY.png	734288	34.653730	34.550937	30.175266	22.269341	8	16	160
BABY.png	764288	34.499533	34.396740	30.021069	23.074223	8	16	160
BABY.png	824288	34.198964	34.096171	29.720500	24.727715	8	16	160

Appendix C: LSB with Dynamic Threshold Algorithm for Gray Scale (LSBDT)

C.1 The main program

% this program was written by Hajer A Al_aswed in 2016-2017
for LSB with Constant threshold algorithm in gray scale [8] and its functions

```
clc
clear
cover_image=imread('C:\Users\hajer\Desktop\thesis\GRAY SCALE\MA for text\home.gif');%Read cover image
[M N L]=size(cover_image);
S=sum(cover_image(:,:));
avg=sum(S)/(M*N);% calculate the mean of cover image
ml=4;
mu=8;
disp('=========================================================')
disp('cover_image  Size_secret_data PSNR  Actual_PSNR   MSNR MSE Ml Mu T ')
disp('              dB          dB                  db ')
p='home.gif';
disp('=========================================================')
start_length =494288;   %start length of secret message
increase_size=30000;
start_length=start_length+increase_size;
Bpp= start_length/M*N;
if (Bpp~=3.14)
 T=160*(3.14-Bpp)+10;
 else
 T=160;
 end
  if (mod(T,16)~=0)
      T=T-mod(T,16);
  end
 Bpp_actual=((T/256)*(log2(ml)))+(((256-T)/256)*(log2(mu)));
 while (Bpp_actual<Bpp)
T=T-mu;
 Bpp_actual=((T/256)*(log2(ml)))+(((256-T)/256)*(log2(mu)));
 end
secret_massege= randi([0 1],1,start_length);%generate the secreat message
[e r]=size(secret_massege);
stego_image=Embedding( ml,mu,T,cover_image,secret_massege); %embedding function
MSE=0;
```

```
for i=1:M
for j=1:N
MSE=MSE+(double(cover_image(i,j))-double(stego_image(i,j)))^2; %calculate the MSE
end
end
MSE=MSE/(M*N);
PSNR=10*log10(255^2/(MSE));%calculate the PSNR
max_value=max(max(cover_image(:,:)));%calculate the maximum actual value
MSNR=10*log10(avg*avg/MSE);%calculate the MSNR
Bpp=(start_length)/(M*N);%calculate the BPP
Actual_PSNR=10*log10(double(max_value)*double(max_value)/MSE);%calculate the APSNR
disp(sprintf('%s %d  %f  %f %f  %f %d  %d %d' ,p,start_length,PSNR,
Actual_PSNR,MSNR,MSE,ml,mu,T));
end
disp('==========================================================')
%show the cover image and stego image
figure
subplot(2,2,[1,3]);
imshow(cover_image);
title('Cover image')
subplot(2,2,[2,4]);
imshow(stego_image);
title('Stego image')
output = extraction( stego_image,T,mu,ml);%extraction function
```

Other functions are the same as shown in Appendices B.2, B.3 for LSBCT

C.2 Screenshots of LSBDT Algorithm for gray scale result for different cover images with size 512×512

Results for Lena image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
lena.BMP	524288	43.203327	42.855845	36.943934	3.109909	4	8	192
lena.BMP	584288	41.739591	41.392109	35.480199	4.356335	4	8	160
lena.BMP	644288	40.031265	39.683783	33.771872	6.455856	4	8	128
lena.BMP	704288	38.852524	38.505042	32.593132	8.468918	4	8	80
lena.BMP	734288	38.336860	37.989378	32.077467	9.536617	4	8	48
lena.BMP	764288	38.042626	37.695144	31.783233	10.205112	4	8	16
lena.BMP	824288	35.636526	35.289045	29.377134	17.759377	8	16	160

Results for Zelda image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
zeldaBMP.bmp	524288	44.133917	43.199824	36.893474	2.510086	4	8	192
zeldaBMP.bmp	584288	42.676404	41.742310	35.435960	3.511074	4	8	160
zeldaBMP.bmp	644288	40.587183	39.653089	33.346739	5.680180	4	8	128
zeldaBMP.bmp	704288	38.972098	38.038004	31.731654	8.238926	4	8	80
zeldaBMP.bmp	734288	38.372418	37.438324	31.131974	9.458855	4	8	48
zeldaBMP.bmp	764288	38.060738	37.126644	30.820294	10.162640	4	8	16
zeldaBMP.bmp	824288	36.424963	35.490869	29.184519	14.810989	8	16	160

Results for Airplane image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
airplane.BMP	524288	40.569041	39.710477	37.503976	5.703957	4	8	192
airplane.BMP	584288	39.791197	38.932633	36.726131	6.822769	4	8	160
airplane.BMP	644288	39.231077	38.372513	36.166011	7.761982	4	8	128
airplane.BMP	704288	38.529906	37.671342	35.464840	9.121994	4	8	80
airplane.BMP	734288	38.284258	37.425694	35.219192	9.652828	4	8	48
airplane.BMP	764288	38.095727	37.237163	35.030662	10.081093	4	8	16
airplane.BMP	824288	33.566819	32.708255	30.501753	28.602139	8	16	160

Results for Baboon image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
Baboon.BMP	524288	43.632592	43.033328	37.723428	2.817223	4	8	192
Baboon.BMP	584288	41.475393	40.876129	35.566230	4.629574	4	8	160
Baboon.BMP	644288	40.001092	39.401827	34.091928	6.500866	4	8	128
Baboon.BMP	704288	38.708346	38.109081	32.799182	8.754791	4	8	80
Baboon.BMP	734288	38.290367	37.691102	32.381203	9.639259	4	8	48
Baboon.BMP	764288	38.038973	37.439708	32.129809	10.213699	4	8	16
Baboon.BMP	824288	35.444204	34.844940	29.535040	18.563503	8	16	160

Results for Barbara image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
barbaraBMP.bmp	524288	42.017444	41.845441	35.934357	4.086353	4	8	192
barbaraBMP.bmp	584288	40.901915	40.729912	34.818828	5.283100	4	8	160
barbaraBMP.bmp	644288	40.067491	39.895488	33.984404	6.402229	4	8	128
barbaraBMP.bmp	704288	39.056830	38.884827	32.973743	8.079739	4	8	80
barbaraBMP.bmp	734288	38.705408	38.533404	32.622320	8.760715	4	8	48
barbaraBMP.bmp	764288	38.108629	37.936626	32.025541	10.051189	4	8	16
barbaraBMP.bmp	824288	34.832107	34.660104	28.749020	21.373203	8	16	160

Results for Elain image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
elainBMP.bmp	524288	42.935040	42.728224	37.497639	3.308083	4	8	192
elainBMP.bmp	584288	41.301537	41.094720	35.864135	4.818665	4	8	160
elainBMP.bmp	644288	39.797366	39.590550	34.359965	6.813084	4	8	128
elainBMP.bmp	704288	38.634517	38.427700	33.197116	8.904892	4	8	80
elainBMP.bmp	734288	38.227739	38.020922	32.790338	9.779270	4	8	48
elainBMP.bmp	764288	38.018896	37.812080	32.581495	10.261024	4	8	16
elainBMP.bmp	824288	35.307156	35.100339	29.869754	19.158646	8	16	160

Results for Goldhill image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
goldhillBMP.bmp	524288	43.168716	42.496152	36.007547	3.134792	4	8	192
goldhillBMP.bmp	584288	42.387492	41.714929	35.226324	3.752590	4	8	160
goldhillBMP.bmp	644288	41.206547	40.533983	34.045378	4.925220	4	8	128
goldhillBMP.bmp	704288	38.992834	38.320271	31.831666	8.199680	4	8	80
goldhillBMP.bmp	734288	38.315076	37.642513	31.153908	9.584572	4	8	48
goldhillBMP.bmp	764288	38.027305	37.354742	30.866136	10.241177	4	8	16
goldhillBMP.bmp	824288	36.355033	35.682469	29.193864	15.051407	8	16	160

Results for Pepper image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
peppersBMP.bmp	524288	43.408087	42.698641	36.882842	2.966686	4	8	192
peppersBMP.bmp	584288	41.417886	40.708440	34.892641	4.691284	4	8	160
peppersBMP.bmp	644288	40.164630	39.455184	33.639385	6.260620	4	8	128
peppersBMP.bmp	704288	38.857559	38.148112	32.332313	8.459106	4	8	80
peppersBMP.bmp	734288	38.427963	37.718516	31.902717	9.338650	4	8	48
peppersBMP.bmp	764288	38.094355	37.384909	31.569110	10.084278	4	8	16
peppersBMP.bmp	824288	35.358311	34.648865	28.833066	18.934299	8	16	160

Results for Lady image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
lady.bmp	524288	42.084938	41.449101	35.411519	4.023338	4	8	192
lady.bmp	584288	40.773181	40.137344	34.099761	5.442047	4	8	160
lady.bmp	644288	40.028353	39.392517	33.354934	6.460186	4	8	128
lady.bmp	704288	39.158543	38.522707	32.485124	7.892708	4	8	80
lady.bmp	734288	38.842559	38.206722	32.169140	8.488373	4	8	48
lady.bmp	764288	38.434155	37.798318	31.760735	9.325344	4	8	16
lady.bmp	824288	34.576679	33.940842	27.903260	22.667961	8	16	160

Results for House image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
house.bmp	524288	43.789080	43.720686	37.318320	2.717518	4	8	192
house.bmp	584288	42.345827	42.277434	35.875068	3.788765	4	8	160
house.bmp	644288	40.160803	40.092409	33.690043	6.266140	4	8	128
house.bmp	704288	38.754594	38.686201	32.283835	8.662056	4	8	80
house.bmp	734288	38.358616	38.290222	31.887856	9.488964	4	8	48
house.bmp	764288	38.077619	38.009226	31.606860	10.123215	4	8	16
house.bmp	824288	36.220871	36.152478	29.750112	15.523628	8	16	160

Results for Home image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
home.bmp	524288	43.787976	43.685183	32.429897	2.718208	4	8	192
home.bmp	584288	43.730398	43.627605	32.372319	2.754486	4	8	160
home.bmp	644288	42.884860	42.782067	31.526781	3.346527	4	8	128
home.bmp	704288	41.044050	40.941257	29.685971	5.112995	4	8	80
home.bmp	734288	39.219629	39.116836	27.861550	7.782471	4	8	48
home.bmp	764288	38.388757	38.285964	27.030678	9.423336	4	8	16
home.bmp	824288	37.776159	37.673366	26.418080	10.850868	8	16	160

Results for Camerman image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
camerman.bmp	524288	43.982946	43.982946	37.312866	2.598877	4	8	192
camerman.bmp	584288	41.026481	41.026481	34.356401	5.133720	4	8	160
camerman.bmp	644288	39.733727	39.733727	33.063646	6.913654	4	8	128
camerman.bmp	704288	38.963994	38.963994	32.293913	8.254314	4	8	80
camerman.bmp	734288	38.758218	38.758218	32.088137	8.654831	4	8	48
camerman.bmp	764288	38.480821	38.480821	31.810740	9.225677	4	8	16
camerman.bmp	824288	35.035104	35.035104	28.365024	20.397171	8	16	160

Results for Boy image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
BOY.bmp	524288	43.269329	43.269329	35.449681	3.063004	4	8	192
BOY.bmp	584288	42.119935	42.119935	34.300287	3.991047	4	8	160
BOY.bmp	644288	41.086417	41.086417	33.266769	5.063358	4	8	128
BOY.bmp	704288	39.289935	39.289935	31.470287	7.657497	4	8	80
BOY.bmp	734288	38.899713	38.899713	31.080066	8.377396	4	8	48
BOY.bmp	764288	38.068592	38.068592	30.248944	10.144279	4	8	16
BOY.bmp	824288	36.035591	36.035591	28.215943	16.200233	8	16	160

Results for Boat image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
boat.bmp	524288	43.652195	43.089350	38.200265	2.804535	4	8	192
boat.bmp	584288	40.515495	39.952649	35.063564	5.774719	4	8	160
boat.bmp	644288	39.476645	38.913799	34.024714	7.335266	4	8	128
boat.bmp	704288	38.823799	38.260953	33.371868	8.525120	4	8	80
boat.bmp	734288	38.481351	37.918505	33.029420	9.224552	4	8	48
boat.bmp	764288	38.041905	37.479059	32.589975	10.206806	4	8	16
boat.bmp	824288	34.321530	33.758685	28.869600	24.039604	8	16	160

Results for Baby image in Figure A.1

cover_image	Size_secret_data	PSNR dB	Actual_PSNR dB	MSNR dB	MSE	Ml	Mu	T
BABY.png	524288	41.819360	41.716568	37.340896	4.277050	4	8	192
BABY.png	584288	40.394614	40.291821	35.916150	5.937710	4	8	160
BABY.png	644288	39.411874	39.309081	34.933410	7.445484	4	8	128
BABY.png	704288	38.680160	38.577367	34.201696	8.811794	4	8	80
BABY.png	734288	38.349848	38.247055	33.871384	9.508141	4	8	48
BABY.png	764288	38.087992	37.985200	33.609528	10.099064	4	8	16
BABY.png	824288	34.217682	34.114890	29.739218	24.621365	8	16	160

Appendix D: LSB Algorithm for Color Images

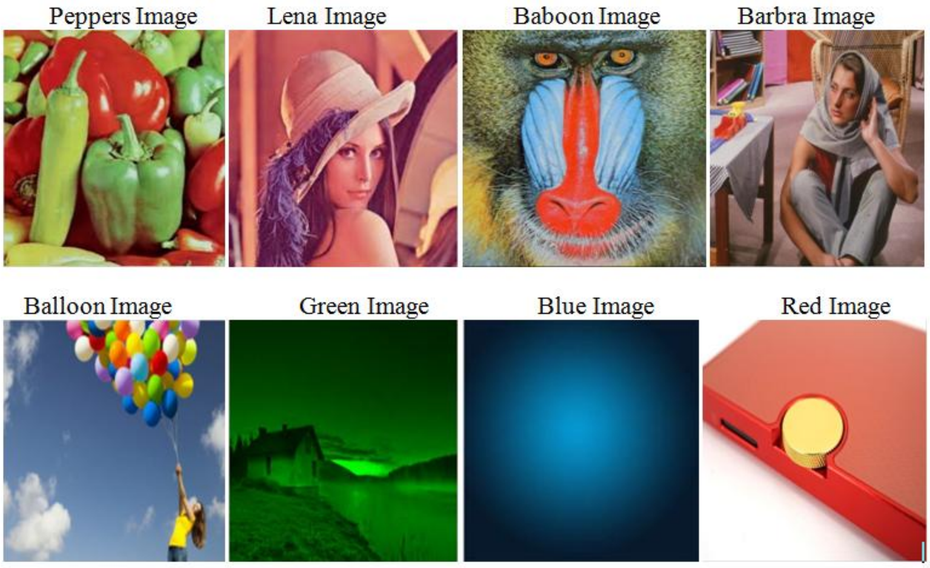


Figure D.1. Color Cover Images Used

D.1 The main program

```
% this program was written by Hajer A Al_aswed in 2016-2017 for LSB with algorithm in color scale and its functions
clc
clear
cover_image=imread('C:\Users\hajer\Desktop\thesis\COLOR SCALE\LSB_color\Balloon (512 x 512).jpg');
disp('=========================================================')
disp('cover_image  Size_secret_data    PSNR    MSE    ')
disp('                                 dB           ')
disp('=========================================================')
p='Balloon_color';
secrt_image=imread('C:\Users\hajer\Desktop\thesis\1.jpg');
[m n l]=size(secrt_image);
k=1;
for i=1:m
for j=1:n
str = dec2bin(secrt_image(i,j),8);
for q=1:8
aa=str2num(str(q));
secret_massege(k)=aa;
k=k+1;
end
```

```
end
end
[e r]=size(secret_massege);
[stego_image,y,u]=Embedding(cover_image,secret_massege);
[MSE_R,MSE_G,MSE_B ] = MSE( y,u,cover_image,stego_image);
mse = (MSE_R + MSE_B + MSE_G)/3;
[PSNR,PSNR_R,PSNR_G,PSNR_B ] = PSNR(mse,MSE_R,MSE_G,MSE_B );
disp(sprintf('%s   %d      %f    %f   ' ,p,r,PSNR,mse));
disp(sprintf('PSNR_R=%f   PSNR_G=%f PSNR_B=%f ',PSNR_R,PSNR_G,PSNR_B));
[MSNR,MSNR_R,MSNR_G,MSNR_B ] = MSNR(cover_image,MSE_R,MSE_G,MSE_B,mse);
disp(sprintf('MSNR_R=%f  MSNR_G=%f  MSNR_B=%f ',MSNR_R,MSNR_G,MSNR_B));
disp(sprintf('MSNR=%f',MSNR));
Actual_PSNR_weight = A_PSNR( cover_image,MSE_R,MSE_G,MSE_B );
disp(sprintf('Actual_PSNR_weight(1/3_1/3_1/3)=%f ',Actual_PSNR_weight));
disp(sprintf('W_PSNR(0.4_0.243_0.357)=%f',(0.4*PSNR_R+0.243*PSNR_G+0.357*PSNR_B))
);
disp(sprintf('W_PSNR(0.4_0.3_0.3)=%f',(0.4*PSNR_R+0.3*PSNR_G+0.3*PSNR_B)));
disp(sprintf('W_PSNR(1/3_1/3_1/3)=%f',(1/3*PSNR_R+1/3*PSNR_G+1/3*PSNR_B)));
disp(sprintf('W_MSNR(0.4_0.243_0.357)=%f',(0.4*MSNR_R+0.243*MSNR_G+0.357*MSNR_
B)));
disp(sprintf('W_MSNR(0.4_0.3_0.3)=%f',(0.4*MSNR_R+0.3*MSNR_G+0.3*MSNR_B)));
disp(sprintf('W_MSNR(1/3_1/3_1/3)=%f',(1/3*MSNR_R+1/3*MSNR_G+1/3*MSNR_B)));
[var_orgR, var_noiseR] = snr(stego_image(:,:,1),cover_image(:,:,1));
[var_orgG, var_noiseG] = snr(stego_image(:,:,2),cover_image(:,:,2));
[var_orgB, var_noiseB] = snr(stego_image(:,:,3),cover_image(:,:,3));
SNR= 10*log10((var_orgR + var_orgG + var_orgB)/(var_noiseR + var_noiseG +   var_noiseB));
disp(sprintf('SNR=%f',SNR));
disp('========================================================')
figure
subplot(2,2,[1,3]);
imshow(cover_image);
title('Cover image')
subplot(2,2,[2,4]);
imshow(stego_image);
title('Stego image')
output=extraction(stego_image);
```

D.2 Embedding function

```
function [ stego_image,y,u ] = Embedding( cover_image,secret_massege)
u=0;
y=0;
[e r]=size(secret_massege);
[M N L]=size(cover_image);
stego_image=cover_image;
k=1;
for i=1:M
for j=1:N
if (k<=r)
EC=3;
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
stego_image(i,j,3)=cover_image(i,j,3)-(mod(cover_image(i,j,3),2^EC))+DEC;
EC=4;
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
stego_image(i,j,2)=cover_image(i,j,2)-(mod(cover_image(i,j,2),2^EC))+DEC;
EC=1;
```

```
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
stego_image(i,j,1)=cover_image(i,j,1)-(mod(cover_image(i,j,1),2^EC))+DEC;
y= i;
u=j;
end
end
end
end
```

D.3 Extraction function

```
function [ output ] = extraction(stego_image )
[M N L]=size(stego_image);
k=1;
kk=1;
for i=1:M
for j=1:N
if (k<=r)
EC=4;
RES=mod(stego_image(i,j,3),2^EC);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
```

```
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
end
if (k<=r)
EC=2;
RES=mod(stego_image(i,j,2),2^EC);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
end
if (k<=r)
EC=2;
RES=mod(stego_image(i,j,1),2^EC);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
```

```
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
end
end
end
end
```

D.4 MSE function

```
function [ MSE_R,MSE_G,MSE_B ] = MSE( y,u,cover_image,stego_image)
[M N L]=size(cover_image);
MSE_R=0;
MSE_G=0;
MSE_B=0;
for i=1:M
for j=1:N
MSE_R=MSE_R+(double(cover_image(i,j,1))-double(stego_image(i,j,1)))^2;
MSE_G=MSE_G+(double(cover_image(i,j,2))-double(stego_image(i,j,2)))^2;
MSE_B=MSE_B+(double(cover_image(i,j,3))-double(stego_image(i,j,3)))^2;
end
end
MSE_R=MSE_R/(((y-1)*512)+u);
MSE_G=MSE_G/(((y-1)*512)+u);
MSE_B=MSE_B/(((y-1)*512)+u);
end
```

D.5 PSNR function

```
function [PSNR,PSNR_R,PSNR_G,PSNR_B ] = PSNR(mse,MSE_R,MSE_G,MSE_B )
PSNR=10*log10(255*255/mse);
PSNR_R=10*log10(255*255/MSE_R);
PSNR_G=10*log10(255*255/MSE_G);
PSNR_B=10*log10(255*255/MSE_B);
end
```

D.6 MSNR function

```
function [ MSNR,MSNR_R,MSNR_G,MSNR_B ] =
MSNR(cover_image,MSE_R,MSE_G,MSE_B,mse)
[M N L]=size(cover_image);
S=sum(cover_image(:,:,1));
avg_R=sum(S)/(M*N);
S2=sum(cover_image(:,:,2));
avg_G=sum(S2)/(M*N);
S3=sum(cover_image(:,:,3));
avg_B=sum(S3)/(M*N);
MSNR_R=10*log10(avg_R*avg_R/MSE_R);
MSNR_G=10*log10(avg_G*avg_G/MSE_G);
MSNR_B=10*log10(avg_B*avg_B/MSE_B);
mean_image=(sum(S2)+sum(S3)+sum(S))/(M*N*L);
MSNR=10*log10(mean_image*mean_image/mse);
end
```

D.7 WAPSNR function

```
function [ Actual_PSNR_weight ]= A_PSNR(cover_image,MSE_R,MSE_G,MSE_B )
max_col_R= max(cover_image(:,:,1));
max_col_G= max(cover_image(:,:,2));
max_col_B= max(cover_image(:,:,3));
max_R= max(max_col_R);
max_G= max(max_col_G);
max_B= max(max_col_B);
Actual_PSNR_R=10*log10(double(max_R)*double(max_R)/MSE_R);
Actual_PSNR_G=10*log10(double(max_G)*double(max_G)/MSE_G);
Actual_PSNR_B=10*log10(double(max_B)*double(max_B)/MSE_B);
Actual_PSNR_weight=1/3*Actual_PSNR_R+1/3*Actual_PSNR_G+1/3*Actual_PSNR_B;
end
```

D.8 SNR function

```
function [var_cover_image, var_noise] = snr(stego_image, cover_image)
[m n l]=size(cover_image);
mean_original = mean(cover_image(:));
tmp= cover_image - mean_original;
var_cover_image  = sum(tmp(:).^2);
var_cover_image  =var_cover_image/(m*n);
noise= stego_image - cover_image;
mean_noise = mean(noise(:));
tmp= noise - mean_noise;
var_noise  = sum(tmp(:).^2);
var_noise  =var_noise/(m*n);
end
```

D.9 Screenshots for LSB for color imges results for different embedding combinations with 8 BPP for cover images with size 512×512

Embedding combination 4_3_1

```
=====================================================================
cover_image        Size_secret_data           PSNR             MSE
                                               dB
=====================================================================
peppers_color             1280000           35.635195           17.764823
PSNR_R=31.829641              PSNR_G=38.074999              PSNR_B=51.178012
MSNR_R=26.823674              MSNR_G=30.931847              MSNR_B=39.118446
MSNR=28.036591
W_PSNR(0.4_0.243_0.357)=40.254631
W_PSNR(0.4_0.3_0.3)=39.507760
W_PSNR(1/3_1/3_1/3)=40.360884
W_MSNR(0.4_0.243_0.357)=32.211194
W_MSNR(0.4_0.3_0.3)=31.744557
W_MSNR(1/3_1/3_1/3)=32.291322
```

```
=====================================================================
cover_image        Size_secret_data           PSNR             MSE
                                               dB
=====================================================================
lena_color             1280000           35.455989           18.513200
PSNR_R=31.609777              PSNR_G=38.063879              PSNR_B=51.148438
MSNR_R=28.590647              MSNR_G=29.851279              MSNR_B=43.487061
MSNR=29.486217
W_PSNR(0.4_0.243_0.357)=40.153426
W_PSNR(0.4_0.3_0.3)=39.407606
W_PSNR(1/3_1/3_1/3)=40.274031
W_MSNR(0.4_0.243_0.357)=34.215000
W_MSNR(0.4_0.3_0.3)=33.437761
W_MSNR(1/3_1/3_1/3)=33.976329
```

```
================================================================================
cover_image         Size_secret_data          PSNR               MSE
                                               dB
================================================================================
baboon_color             1280000            35.634201         17.768890
PSNR_R=31.852268            PSNR_G=37.976842               PSNR_B=51.15714
MSNR_R=25.978631            MSNR_G=31.502331               MSNR_B=43.49189
MSNR=28.994603
W_PSNR(0.4_0.243_0.357)=40.232381
W_PSNR(0.4_0.3_0.3)=39.481104
W_PSNR(1/3_1/3_1/3)=40.328752
W_MSNR(0.4_0.243_0.357)=33.573124
W_MSNR(0.4_0.3_0.3)=32.889719
W_MSNR(1/3_1/3_1/3)=33.657618
```

```
================================================================================
cover_image         Size_secret_data          PSNR               MSE
                                               dB
================================================================================
barbra_color             1280000            35.629175         17.789465
PSNR_R=31.840527            PSNR_G=37.999381               PSNR_B=51.153226
MSNR_R=26.291158            MSNR_G=30.051279               MSNR_B=42.437818
MSNR=28.332144
W_PSNR(0.4_0.243_0.357)=40.231762
W_PSNR(0.4_0.3_0.3)=39.481993
W_PSNR(1/3_1/3_1/3)=40.331045
W_MSNR(0.4_0.243_0.357)=32.969225
W_MSNR(0.4_0.3_0.3)=32.263192
W_MSNR(1/3_1/3_1/3)=32.926752
```

```
================================================================================
cover_image         Size_secret_data          PSNR               MSE
                                               dB
================================================================================
Balloon_color            1280000            35.637901         17.753756
PSNR_R=31.860033            PSNR_G=37.963387               PSNR_B=51.172212
MSNR_R=24.409883            MSNR_G=31.737286               MSNR_B=46.074971
MSNR=29.433037
W_PSNR(0.4_0.243_0.357)=40.237596
W_PSNR(0.4_0.3_0.3)=39.484693
W_PSNR(1/3_1/3_1/3)=40.331877
W_MSNR(0.4_0.243_0.357)=33.924878
W_MSNR(0.4_0.3_0.3)=33.107630
W_MSNR(1/3_1/3_1/3)=34.074047
```

```
================================================================================
cover_image         Size_secret_data          PSNR               MSE
                                               dB
================================================================================
Blue_color               1280000            37.438532         11.728085
PSNR_R=34.260334            PSNR_G=38.003657               PSNR_B=51.085706
MSNR_R=3.226578             MSNR_G=25.739273               MSNR_B=42.214812
MSNR=23.908515
W_PSNR(0.4_0.243_0.357)=41.176619
W_PSNR(0.4_0.3_0.3)=40.430942
W_PSNR(1/3_1/3_1/3)=41.116566
W_MSNR(0.4_0.243_0.357)=22.615963
W_MSNR(0.4_0.3_0.3)=21.676857
W_MSNR(1/3_1/3_1/3)=23.726888
```

```
================================================================================
cover_image          Size_secret_data            PSNR               MSE
                                                  dB
================================================================================
green_color              1280000              34.097726          25.310910
PSNR_R=29.949828              PSNR_G=38.261771              PSNR_B=51.606104
MSNR_R=-15.897324             MSNR_G=27.750596              MSNR_B=6.221717
MSNR=14.344211
W_PSNR(0.4_0.243_0.357)=39.700921
W_PSNR(0.4_0.3_0.3)=38.940294
W_PSNR(1/3_1/3_1/3)=39.939234
W_MSNR(0.4_0.243_0.357)=2.605618
W_MSNR(0.4_0.3_0.3)=3.832764
W_MSNR(1/3_1/3_1/3)=6.024996
```

```
================================================================================
cover_image          Size_secret_data            PSNR               MSE
                                                  dB
================================================================================
red_color                1280000              34.981288          20.651498
PSNR_R=31.090977              PSNR_G=37.771933              PSNR_B=51.032912
MSNR_R=29.951912              MSNR_G=32.786249              MSNR_B=44.970399
MSNR=31.185747
W_PSNR(0.4_0.243_0.357)=39.833720
W_PSNR(0.4_0.3_0.3)=39.077844
W_PSNR(1/3_1/3_1/3)=39.965274
W_MSNR(0.4_0.243_0.357)=36.002256
W_MSNR(0.4_0.3_0.3)=35.307759
W_MSNR(1/3_1/3_1/3)=35.902853
```

Embedding combination 4_1_3

```
================================================================================
cover_image          Size_secret_data            PSNR               MSE
                                                  dB
================================================================================
peppers_color            1280000              35.754077          17.285131
PSNR_R=31.829641              PSNR_G=51.139150              PSNR_B=38.742708
MSNR_R=26.823674              MSNR_G=43.995998              MSNR_B=26.683142
MSNR=28.155473
W_PSNR(0.4_0.243_0.357)=38.989817
W_PSNR(0.4_0.3_0.3)=39.696414
W_PSNR(1/3_1/3_1/3)=40.570500
W_MSNR(0.4_0.243_0.357)=30.946379
W_MSNR(0.4_0.3_0.3)=31.933212
W_MSNR(1/3_1/3_1/3)=32.500938
```

```
================================================================================
cover_image          Size_secret_data            PSNR               MSE
                                                  dB
================================================================================
lena_color               1280000              35.565104          18.053854
PSNR_R=31.609777              PSNR_G=51.157257              PSNR_B=38.696715
MSNR_R=28.590647              MSNR_G=42.944657              MSNR_B=31.035337
MSNR=29.595333
W_PSNR(0.4_0.243_0.357)=38.889851
W_PSNR(0.4_0.3_0.3)=39.600102
W_PSNR(1/3_1/3_1/3)=40.487916
W_MSNR(0.4_0.243_0.357)=32.951426
W_MSNR(0.4_0.3_0.3)=33.630257
W_MSNR(1/3_1/3_1/3)=34.190214
```

```
=============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
=============================================================================
baboon_color            1280000                 35.770332          17.220556
PSNR_R=31.852268            PSNR_G=51.147895                 PSNR_B=38.728226
MSNR_R=25.978631            MSNR_G=44.673384                 MSNR_B=31.062970
MSNR=29.130734
W_PSNR(0.4_0.243_0.357)=38.995822
W_PSNR(0.4_0.3_0.3)=39.703743
W_PSNR(1/3_1/3_1/3)=40.576130
W_MSNR(0.4_0.243_0.357)=32.336565
W_MSNR(0.4_0.3_0.3)=33.112359
W_MSNR(1/3_1/3_1/3)=33.904995
```

```
=============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
=============================================================================
barbra_color            1280000                 35.760553          17.259373
PSNR_R=31.840527            PSNR_G=51.139041                 PSNR_B=38.727965
MSNR_R=26.291158            MSNR_G=43.190940                 MSNR_B=30.012556
MSNR=28.463523
W_PSNR(0.4_0.243_0.357)=38.988881
W_PSNR(0.4_0.3_0.3)=39.696313
W_PSNR(1/3_1/3_1/3)=40.569178
W_MSNR(0.4_0.243_0.357)=31.726344
W_MSNR(0.4_0.3_0.3)=32.477512
W_MSNR(1/3_1/3_1/3)=33.164885
```

```
=============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
=============================================================================
Blue_color            1280000                 37.615874          11.258823
PSNR_R=34.260334            PSNR_G=51.171884                 PSNR_B=38.637266
MSNR_R=3.226578            MSNR_G=38.907500                 MSNR_B=29.766373
MSNR=24.085856
W_PSNR(0.4_0.243_0.357)=39.932406
W_PSNR(0.4_0.3_0.3)=40.646879
W_PSNR(1/3_1/3_1/3)=41.356495
W_MSNR(0.4_0.243_0.357)=21.371749
W_MSNR(0.4_0.3_0.3)=21.892793
W_MSNR(1/3_1/3_1/3)=23.966817
```

```
=============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
=============================================================================
green_color            1280000                 34.235772          24.519023
PSNR_R=29.949828            PSNR_G=51.103376                 PSNR_B=39.514179
MSNR_R=-15.897324            MSNR_G=40.592200                 MSNR_B=-5.870208
MSNR=14.482257
W_PSNR(0.4_0.243_0.357)=38.504613
W_PSNR(0.4_0.3_0.3)=39.165198
W_PSNR(1/3_1/3_1/3)=40.189128
W_MSNR(0.4_0.243_0.357)=1.409311
W_MSNR(0.4_0.3_0.3)=4.057668
W_MSNR(1/3_1/3_1/3)=6.274890
```

```
=============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
=============================================================================
red_color            1280000                 35.129184          19.960065
PSNR_R=31.090977            PSNR_G=51.147351                 PSNR_B=38.685749
MSNR_R=29.951912            MSNR_G=46.161667                 MSNR_B=32.623236
MSNR=31.333643
W_PSNR(0.4_0.243_0.357)=38.676009
W_PSNR(0.4_0.3_0.3)=39.386321
W_PSNR(1/3_1/3_1/3)=40.308025
W_MSNR(0.4_0.243_0.357)=34.844545
W_MSNR(0.4_0.3_0.3)=35.616236
W_MSNR(1/3_1/3_1/3)=36.245605
```

Embedding combination 3_4_1

```
==========================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==========================================================================
peppers_color              1280000                  35.675737           17.599756
PSNR_R=37.902943                PSNR_G=31.922697                PSNR_B=51.178012
MSNR_R=32.896977                MSNR_G=24.779545                MSNR_B=39.118446
MSNR=28.077134
W_PSNR(0.4_0.243_0.357)=41.188943
W_PSNR(0.4_0.3_0.3)=40.091390
W_PSNR(1/3_1/3_1/3)=40.334551
W_MSNR(0.4_0.243_0.357)=33.145505
W_MSNR(0.4_0.3_0.3)=32.328188
W_MSNR(1/3_1/3_1/3)=32.264989
```

```
==========================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==========================================================================
lena_color              1280000                  35.679372           17.585033
PSNR_R=37.902253                PSNR_G=31.927819                PSNR_B=51.148438
MSNR_R=34.883123                MSNR_G=23.715220                MSNR_B=43.487061
MSNR=29.709600
W_PSNR(0.4_0.243_0.357)=41.179354
W_PSNR(0.4_0.3_0.3)=40.083779
W_PSNR(1/3_1/3_1/3)=40.326170
W_MSNR(0.4_0.243_0.357)=35.240928
W_MSNR(0.4_0.3_0.3)=34.113933
W_MSNR(1/3_1/3_1/3)=34.028468
```

```
==========================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==========================================================================
baboon_color              1280000                  35.655562           17.681706
PSNR_R=37.893077                PSNR_G=31.899933                PSNR_B=51.157148
MSNR_R=32.019441                MSNR_G=25.425422                MSNR_B=43.491891
MSNR=29.015964
W_PSNR(0.4_0.243_0.357)=41.172016
W_PSNR(0.4_0.3_0.3)=40.074355
W_PSNR(1/3_1/3_1/3)=40.316719
W_MSNR(0.4_0.243_0.357)=34.512759
W_MSNR(0.4_0.3_0.3)=33.482970
W_MSNR(1/3_1/3_1/3)=33.645585
```

```
==========================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==========================================================================
barbra_color              1280000                  35.663901           17.647785
PSNR_R=37.894432                PSNR_G=31.910177                PSNR_B=51.153226
MSNR_R=32.345063                MSNR_G=23.962075                MSNR_B=42.437818
MSNR=28.366871
W_PSNR(0.4_0.243_0.357)=41.173648
W_PSNR(0.4_0.3_0.3)=40.076794
W_PSNR(1/3_1/3_1/3)=40.319278
W_MSNR(0.4_0.243_0.357)=33.911110
W_MSNR(0.4_0.3_0.3)=32.857993
W_MSNR(1/3_1/3_1/3)=32.914985
```

```
==========================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
==========================================================================
Balloon_color              1280000               35.663392         17.649856
PSNR_R=37.946121              PSNR_G=31.896376                  PSNR_B=51.172212
MSNR_R=30.495971              MSNR_G=25.670275                  MSNR_B=46.074971
MSNR=29.458527
W_PSNR(0.4_0.243_0.357)=41.197748
W_PSNR(0.4_0.3_0.3)=40.099025
W_PSNR(1/3_1/3_1/3)=40.338236
W_MSNR(0.4_0.243_0.357)=34.885030
W_MSNR(0.4_0.3_0.3)=33.721962
W_MSNR(1/3_1/3_1/3)=34.080406
```

```
==========================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
==========================================================================
Blue_color              1280000               35.519825         18.243069
PSNR_R=36.779781              PSNR_G=32.048353                  PSNR_B=51.085706
MSNR_R=5.746025             MSNR_G=19.783968                  MSNR_B=42.214812
MSNR=21.989807
W_PSNR(0.4_0.243_0.357)=40.737259
W_PSNR(0.4_0.3_0.3)=39.652130
W_PSNR(1/3_1/3_1/3)=39.971280
W_MSNR(0.4_0.243_0.357)=22.176603
W_MSNR(0.4_0.3_0.3)=20.898044
W_MSNR(1/3_1/3_1/3)=22.581602
```

```
==========================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
==========================================================================
green_color              1280000               35.583683         17.976785
PSNR_R=36.849506              PSNR_G=32.104837                  PSNR_B=51.606104
MSNR_R=-8.997645             MSNR_G=21.593661                  MSNR_B=6.221717
MSNR=15.830168
W_PSNR(0.4_0.243_0.357)=40.964657
W_PSNR(0.4_0.3_0.3)=39.853085
W_PSNR(1/3_1/3_1/3)=40.186816
W_MSNR(0.4_0.243_0.357)=3.869354
W_MSNR(0.4_0.3_0.3)=4.745555
W_MSNR(1/3_1/3_1/3)=6.272578
```

```
==========================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
==========================================================================
red_color              1280000               35.701765         17.494594
PSNR_R=37.613280              PSNR_G=32.034260                  PSNR_B=51.032912
MSNR_R=36.474215              MSNR_G=27.048576                  MSNR_B=44.970399
MSNR=31.906224
W_PSNR(0.4_0.243_0.357)=41.048386
W_PSNR(0.4_0.3_0.3)=39.965463
W_PSNR(1/3_1/3_1/3)=40.226817
W_MSNR(0.4_0.243_0.357)=37.216922
W_MSNR(0.4_0.3_0.3)=36.195378
W_MSNR(1/3_1/3_1/3)=36.164397
```

Embedding combination 3_1_4

```
==========================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
==========================================================================
peppers_color              1280000               36.349222         15.071558
PSNR_R=37.902943              PSNR_G=51.120740                  PSNR_B=32.793890
MSNR_R=32.896977              MSNR_G=43.977588                  MSNR_B=20.734324
MSNR=28.750619
W_PSNR(0.4_0.243_0.357)=39.290936
W_PSNR(0.4_0.3_0.3)=40.335566
W_PSNR(1/3_1/3_1/3)=40.605858
W_MSNR(0.4_0.243_0.357)=31.247498
W_MSNR(0.4_0.3_0.3)=32.572364
W_MSNR(1/3_1/3_1/3)=32.536296
```

```
==========================================================================
cover_image          Size_secret_data             PSNR              MSE
                                                   dB
==========================================================================
lena_color               1280000                 36.048519         16.152081
PSNR_R=37.902253               PSNR_G=51.138770                PSNR_B=32.400274
MSNR_R=34.883123               MSNR_G=42.926170                MSNR_B=24.738896
MSNR=30.078747
W_PSNR(0.4_0.243_0.357)=39.154520
W_PSNR(0.4_0.3_0.3)=40.222614
W_PSNR(1/3_1/3_1/3)=40.480432
W_MSNR(0.4_0.243_0.357)=33.216095
W_MSNR(0.4_0.3_0.3)=34.252769
W_MSNR(1/3_1/3_1/3)=34.182730
```

```
==========================================================================
cover_image          Size_secret_data             PSNR              MSE
                                                   dB
==========================================================================
baboon_color               1280000                 36.324166         15.158763
PSNR_R=37.893077               PSNR_G=51.142895                PSNR_B=32.763471
MSNR_R=32.019441               MSNR_G=44.668385                MSNR_B=25.098215
MSNR=29.684568
W_PSNR(0.4_0.243_0.357)=39.281514
W_PSNR(0.4_0.3_0.3)=40.329141
W_PSNR(1/3_1/3_1/3)=40.599815
W_MSNR(0.4_0.243_0.357)=32.622257
W_MSNR(0.4_0.3_0.3)=33.737756
W_MSNR(1/3_1/3_1/3)=33.928680
```

```
==========================================================================
cover_image          Size_secret_data             PSNR              MSE
                                                   dB
==========================================================================
barbra_color               1280000                 36.301955         15.236488
PSNR_R=37.894432               PSNR_G=51.139801                PSNR_B=32.733776
MSNR_R=32.345063               MSNR_G=43.191699                MSNR_B=24.018367
MSNR=29.004924
W_PSNR(0.4_0.243_0.357)=39.270702
W_PSNR(0.4_0.3_0.3)=40.319846
W_PSNR(1/3_1/3_1/3)=40.589336
W_MSNR(0.4_0.243_0.357)=32.008165
W_MSNR(0.4_0.3_0.3)=33.101045
W_MSNR(1/3_1/3_1/3)=33.185043
```

```
==========================================================================
cover_image          Size_secret_data             PSNR              MSE
                                                   dB
==========================================================================
Balloon_color               1280000                 36.397708         14.904229
PSNR_R=37.946121               PSNR_G=51.140289                PSNR_B=32.844442
MSNR_R=30.495971               MSNR_G=44.914189                MSNR_B=27.747200
MSNR=30.192844
W_PSNR(0.4_0.243_0.357)=39.331004
W_PSNR(0.4_0.3_0.3)=40.373868
W_PSNR(1/3_1/3_1/3)=40.643617
W_MSNR(0.4_0.243_0.357)=33.018287
W_MSNR(0.4_0.3_0.3)=33.996805
W_MSNR(1/3_1/3_1/3)=34.385787
```

```
==========================================================================
cover_image          Size_secret_data             PSNR              MSE
                                                   dB
==========================================================================
Blue_color               1280000                 35.808325         17.070563
PSNR_R=36.779781               PSNR_G=51.165110                PSNR_B=32.441117
MSNR_R=5.746025               MSNR_G=38.900726                MSNR_B=23.570223
MSNR=22.278307
W_PSNR(0.4_0.243_0.357)=38.726513
W_PSNR(0.4_0.3_0.3)=39.793781
W_PSNR(1/3_1/3_1/3)=40.128669
W_MSNR(0.4_0.243_0.357)=20.165856
W_MSNR(0.4_0.3_0.3)=21.039695
W_MSNR(1/3_1/3_1/3)=22.738992
```

```
================================================================================
cover_image         Size_secret_data            PSNR                MSE
                                                  dB
================================================================================
green_color              1280000                35.990377           16.369775
PSNR_R=36.849506              PSNR_G=51.114368                 PSNR_B=32.668515
MSNR_R=-8.997645              MSNR_G=40.603193                 MSNR_B=-12.715872
MSNR=16.236861
W_PSNR(0.4_0.243_0.357)=38.823254
W_PSNR(0.4_0.3_0.3)=39.874667
W_PSNR(1/3_1/3_1/3)=40.210796
W_MSNR(0.4_0.243_0.357)=1.727951
W_MSNR(0.4_0.3_0.3)=4.767138
W_MSNR(1/3_1/3_1/3)=6.296558
```

```
================================================================================
cover_image         Size_secret_data            PSNR                MSE
                                                  dB
================================================================================
red_color                1280000                35.901220           16.709304
PSNR_R=37.613280              PSNR_G=51.137902                 PSNR_B=32.291792
MSNR_R=36.474215              MSNR_G=46.152217                 MSNR_B=26.229280
MSNR=32.105679
W_PSNR(0.4_0.243_0.357)=38.999992
W_PSNR(0.4_0.3_0.3)=40.074220
W_PSNR(1/3_1/3_1/3)=40.347658
W_MSNR(0.4_0.243_0.357)=35.168528
W_MSNR(0.4_0.3_0.3)=36.304135
W_MSNR(1/3_1/3_1/3)=36.285237
```

Embedding combination 1_3_4

```
================================================================================
cover_image         Size_secret_data            PSNR                MSE
                                                  dB
================================================================================
peppers_color            1280000                36.358849           15.038188
PSNR_R=51.140669              PSNR_G=37.943440                 PSNR_B=32.793890
MSNR_R=46.134703              MSNR_G=30.800288                 MSNR_B=20.734324
MSNR=28.760245
W_PSNR(0.4_0.243_0.357)=41.383942
W_PSNR(0.4_0.3_0.3)=41.677467
W_PSNR(1/3_1/3_1/3)=40.626000
W_MSNR(0.4_0.243_0.357)=33.340505
W_MSNR(0.4_0.3_0.3)=33.914265
W_MSNR(1/3_1/3_1/3)=32.556438
```

```
================================================================================
cover_image         Size_secret_data            PSNR                MSE
                                                  dB
================================================================================
lena_color               1280000                36.058180           16.116190
PSNR_R=51.141538              PSNR_G=37.946714                 PSNR_B=32.400274
MSNR_R=48.122408              MSNR_G=29.734115                 MSNR_B=24.738896
MSNR=30.088409
W_PSNR(0.4_0.243_0.357)=41.244564
W_PSNR(0.4_0.3_0.3)=41.560711
W_PSNR(1/3_1/3_1/3)=40.496175
W_MSNR(0.4_0.243_0.357)=35.306139
W_MSNR(0.4_0.3_0.3)=35.590866
W_MSNR(1/3_1/3_1/3)=34.198473
```

```
-----------------------------------------------------------------------------
cover_image        Size_secret_data          PSNR              MSE
                                              dB
=============================================================================
baboon_color             1280000               36.327248         15.148008
PSNR_R=51.127499             PSNR_G=37.907095               PSNR_B=32.763471
MSNR_R=45.253863             MSNR_G=31.432584               MSNR_B=25.098215
MSNR=29.687650
W_PSNR(0.4_0.243_0.357)=41.358983
W_PSNR(0.4_0.3_0.3)=41.652169
W_PSNR(1/3_1/3_1/3)=40.599355
W_MSNR(0.4_0.243_0.357)=34.699726
W_MSNR(0.4_0.3_0.3)=35.060785
W_MSNR(1/3_1/3_1/3)=33.928220
```

```
=============================================================================
cover_image        Size_secret_data          PSNR              MSE
                                              dB
=============================================================================
barbra_color             1280000               36.309763         15.209121
PSNR_R=51.133944             PSNR_G=37.928610               PSNR_B=32.733776
MSNR_R=45.584575             MSNR_G=29.980509               MSNR_B=24.018367
MSNR=29.012732
W_PSNR(0.4_0.243_0.357)=41.356188
W_PSNR(0.4_0.3_0.3)=41.652293
W_PSNR(1/3_1/3_1/3)=40.598777
W_MSNR(0.4_0.243_0.357)=34.093651
W_MSNR(0.4_0.3_0.3)=34.433493
W_MSNR(1/3_1/3_1/3)=33.194484
```

```
=============================================================================
cover_image        Size_secret_data          PSNR              MSE
                                              dB
=============================================================================
Balloon_color            1280000               36.381057         14.961483
PSNR_R=51.149635             PSNR_G=37.874773               PSNR_B=32.844442
MSNR_R=43.699485             MSNR_G=31.648673               MSNR_B=27.747200
MSNR=30.176193
W_PSNR(0.4_0.243_0.357)=41.388890
W_PSNR(0.4_0.3_0.3)=41.675618
W_PSNR(1/3_1/3_1/3)=40.622950
W_MSNR(0.4_0.243_0.357)=35.076172
W_MSNR(0.4_0.3_0.3)=35.298556
W_MSNR(1/3_1/3_1/3)=34.365119
```

```
=============================================================================
cover_image        Size_secret_data          PSNR              MSE
                                              dB
=============================================================================
Blue_color               1280000               36.084681         16.018146
PSNR_R=51.154424             PSNR_G=37.922813               PSNR_B=32.441117
MSNR_R=20.120668             MSNR_G=25.658429               MSNR_B=23.570223
MSNR=22.554663
W_PSNR(0.4_0.243_0.357)=41.258492
W_PSNR(0.4_0.3_0.3)=41.570949
W_PSNR(1/3_1/3_1/3)=40.506118
W_MSNR(0.4_0.243_0.357)=22.697835
W_MSNR(0.4_0.3_0.3)=22.816863
W_MSNR(1/3_1/3_1/3)=23.116440
```

```
=============================================================================
cover_image        Size_secret_data          PSNR              MSE
                                              dB
=============================================================================
green_color              1280000               36.290224         15.277700
PSNR_R=51.137305             PSNR_G=38.062684               PSNR_B=32.668515
MSNR_R=5.290154              MSNR_G=27.551509               MSNR_B=-12.715872
MSNR=16.536709
W_PSNR(0.4_0.243_0.357)=41.366814
W_PSNR(0.4_0.3_0.3)=41.674282
W_PSNR(1/3_1/3_1/3)=40.622835
W_MSNR(0.4_0.243_0.357)=4.271512
W_MSNR(0.4_0.3_0.3)=6.566753
W_MSNR(1/3_1/3_1/3)=6.708597
```

```
===========================================================================
cover_image          Size_secret_data             PSNR              MSE
                                                   dB
===========================================================================
red_color              1280000                  35.913842          16.660813
PSNR_R=51.141321              PSNR_G=37.669573                  PSNR_B=32.291792
MSNR_R=50.002256              MSNR_G=32.683888                  MSNR_B=26.229280
MSNR=32.118301
W_PSNR(0.4_0.243_0.357)=41.138404
W_PSNR(0.4_0.3_0.3)=41.444938
W_PSNR(1/3_1/3_1/3)=40.367562
W_MSNR(0.4_0.243_0.357)=37.306940
W_MSNR(0.4_0.3_0.3)=37.674853
W_MSNR(1/3_1/3_1/3)=36.305141
```

Embedding combination 1_4_3

```
===========================================================================
cover_image          Size_secret_data             PSNR              MSE
                                                   dB
===========================================================================
peppers_color              1280000                35.583196          17.978802
PSNR_R=51.140669              PSNR_G=31.622819                  PSNR_B=38.742708
MSNR_R=46.134703              MSNR_G=24.479667                  MSNR_B=26.683142
MSNR=27.984593
W_PSNR(0.4_0.243_0.357)=41.971760
W_PSNR(0.4_0.3_0.3)=41.565926
W_PSNR(1/3_1/3_1/3)=40.502065
W_MSNR(0.4_0.243_0.357)=33.928322
W_MSNR(0.4_0.3_0.3)=33.802724
W_MSNR(1/3_1/3_1/3)=32.432504
```

```
===========================================================================
cover_image          Size_secret_data             PSNR              MSE
                                                   dB
===========================================================================
lena_color             1280000              35.594354          17.932671
PSNR_R=51.141538              PSNR_G=31.645272                  PSNR_B=38.696715
MSNR_R=48.122408              MSNR_G=23.432673                  MSNR_B=31.035337
MSNR=29.624583
W_PSNR(0.4_0.243_0.357)=41.961144
W_PSNR(0.4_0.3_0.3)=41.559211
W_PSNR(1/3_1/3_1/3)=40.494508
W_MSNR(0.4_0.243_0.357)=36.022718
W_MSNR(0.4_0.3_0.3)=35.589366
W_MSNR(1/3_1/3_1/3)=34.196806
```

```
===========================================================================
cover_image          Size_secret_data             PSNR              MSE
                                                   dB
===========================================================================
baboon_color             1280000                35.621688          17.820160
PSNR_R=51.127499              PSNR_G=31.672249                  PSNR_B=38.728226
MSNR_R=45.253863              MSNR_G=25.197739                  MSNR_B=31.062970
MSNR=28.982089
W_PSNR(0.4_0.243_0.357)=41.973333
W_PSNR(0.4_0.3_0.3)=41.571142
W_PSNR(1/3_1/3_1/3)=40.509325
W_MSNR(0.4_0.243_0.357)=35.314076
W_MSNR(0.4_0.3_0.3)=34.979758
W_MSNR(1/3_1/3_1/3)=33.838190
```

```
==========================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==========================================================================
barbra_color               1280000                  35.615546          17.845377
PSNR_R=51.133944               PSNR_G=31.664809                    PSNR_B=38.727965
MSNR_R=45.584575               MSNR_G=23.716707                    MSNR_B=30.012556
MSNR=28.318516
W_PSNR(0.4_0.243_0.357)=41.974009
W_PSNR(0.4_0.3_0.3)=41.571409
W_PSNR(1/3_1/3_1/3)=40.508906
W_MSNR(0.4_0.243_0.357)=34.711472
W_MSNR(0.4_0.3_0.3)=34.352609
W_MSNR(1/3_1/3_1/3)=33.104613
```

```
==========================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==========================================================================
Balloon_color               1280000                  35.627387          17.796790
PSNR_R=51.149635               PSNR_G=31.671862                    PSNR_B=38.764011
MSNR_R=43.699485               MSNR_G=25.445761                    MSNR_B=33.666769
MSNR=29.422522
W_PSNR(0.4_0.243_0.357)=41.994868
W_PSNR(0.4_0.3_0.3)=41.590616
W_PSNR(1/3_1/3_1/3)=40.528502
W_MSNR(0.4_0.243_0.357)=35.682151
W_MSNR(0.4_0.3_0.3)=35.213553
W_MSNR(1/3_1/3_1/3)=34.270672
```

```
==========================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==========================================================================
Blue_color              1280000                  35.663439          17.649665
PSNR_R=51.154424               PSNR_G=31.740795                    PSNR_B=38.637266
MSNR_R=20.120668               MSNR_G=19.476411                    MSNR_B=29.766373
MSNR=22.133421
W_PSNR(0.4_0.243_0.357)=41.968287
W_PSNR(0.4_0.3_0.3)=41.575188
W_PSNR(1/3_1/3_1/3)=40.510829
W_MSNR(0.4_0.243_0.357)=23.407630
W_MSNR(0.4_0.3_0.3)=22.821103
W_MSNR(1/3_1/3_1/3)=23.121151
```

```
==========================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==========================================================================
green_color             1280000                  35.821866          17.017423
PSNR_R=51.137305               PSNR_G=31.767962                    PSNR_B=39.514179
MSNR_R=5.290154              MSNR_G=21.256787                    MSNR_B=-5.870208
MSNR=16.068351
W_PSNR(0.4_0.243_0.357)=42.281099
W_PSNR(0.4_0.3_0.3)=41.839565
W_PSNR(1/3_1/3_1/3)=40.806482
W_MSNR(0.4_0.243_0.357)=5.185796
W_MSNR(0.4_0.3_0.3)=6.732035
W_MSNR(1/3_1/3_1/3)=6.892244
```

```
==========================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==========================================================================
red_color             1280000                  35.552818          18.105002
PSNR_R=51.141321               PSNR_G=31.597273                    PSNR_B=38.685749
MSNR_R=50.002256               MSNR_G=26.611588                    MSNR_B=32.623236
MSNR=31.757277
W_PSNR(0.4_0.243_0.357)=41.945478
W_PSNR(0.4_0.3_0.3)=41.541435
W_PSNR(1/3_1/3_1/3)=40.474781
W_MSNR(0.4_0.243_0.357)=38.114014
W_MSNR(0.4_0.3_0.3)=37.771350
W_MSNR(1/3_1/3_1/3)=36.412360
```

Embedding combination 2_2_4

```
==========================================================================
cover_image        Size_secret_data            PSNR              MSE
                                                dB
==========================================================================
peppers_color              1280000               36.971273          13.060298
PSNR_R=44.153184              PSNR_G=44.137063                PSNR_B=32.793890
MSNR_R=39.147218              MSNR_G=36.993911                MSNR_B=20.734324
MSNR=29.372669
W_PSNR(0.4_0.243_0.357)=40.093999
W_PSNR(0.4_0.3_0.3)=40.740560
W_PSNR(1/3_1/3_1/3)=40.361379
W_MSNR(0.4_0.243_0.357)=32.050561
W_MSNR(0.4_0.3_0.3)=32.977358
W_MSNR(1/3_1/3_1/3)=32.291818
```

```
==========================================================================
cover_image        Size_secret_data            PSNR              MSE
                                                dB
==========================================================================
lena_color              1280000             36.626284          14.140083
PSNR_R=44.138741              PSNR_G=44.156053                PSNR_B=32.400274
MSNR_R=41.119610              MSNR_G=35.943453                MSNR_B=24.738896
MSNR=30.656513
W_PSNR(0.4_0.243_0.357)=39.952315
W_PSNR(0.4_0.3_0.3)=40.622394
W_PSNR(1/3_1/3_1/3)=40.231689
W_MSNR(0.4_0.243_0.357)=34.013889
W_MSNR(0.4_0.3_0.3)=34.652549
W_MSNR(1/3_1/3_1/3)=33.933987
```

```
==========================================================================
cover_image        Size_secret_data            PSNR              MSE
                                                dB
==========================================================================
barbra_color              1280000               36.918895          13.218765
PSNR_R=44.137269              PSNR_G=44.154564                PSNR_B=32.733776
MSNR_R=38.587899              MSNR_G=36.206463                MSNR_B=24.018367
MSNR=29.621864
W_PSNR(0.4_0.243_0.357)=40.070425
W_PSNR(0.4_0.3_0.3)=40.721409
W_PSNR(1/3_1/3_1/3)=40.341870
W_MSNR(0.4_0.243_0.357)=32.807887
W_MSNR(0.4_0.3_0.3)=33.502609
W_MSNR(1/3_1/3_1/3)=32.937577
```


```
==========================================================================
cover_image        Size_secret_data            PSNR              MSE
                                                dB
==========================================================================
lena_color              1280000             36.626284          14.140083
PSNR_R=44.138741              PSNR_G=44.156053                PSNR_B=32.400274
MSNR_R=41.119610              MSNR_G=35.943453                MSNR_B=24.738896
MSNR=30.656513
W_PSNR(0.4_0.243_0.357)=39.952315
W_PSNR(0.4_0.3_0.3)=40.622394
W_PSNR(1/3_1/3_1/3)=40.231689
W_MSNR(0.4_0.243_0.357)=34.013889
W_MSNR(0.4_0.3_0.3)=34.652549
W_MSNR(1/3_1/3_1/3)=33.933987
```


```
==========================================================================
cover_image        Size_secret_data            PSNR              MSE
                                                dB
==========================================================================
Balloon_color              1280000               37.014173          12.931923
PSNR_R=44.150850              PSNR_G=44.121497                PSNR_B=32.844442
MSNR_R=36.700700              MSNR_G=37.895397                MSNR_B=27.747200
MSNR=30.809308
W_PSNR(0.4_0.243_0.357)=40.107329
W_PSNR(0.4_0.3_0.3)=40.750122
W_PSNR(1/3_1/3_1/3)=40.372263
W_MSNR(0.4_0.243_0.357)=33.794612
W_MSNR(0.4_0.3_0.3)=34.373059
W_MSNR(1/3_1/3_1/3)=34.114432
```

```
===============================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
===============================================================================
Blue_color              1280000                36.658028           14.037106
PSNR_R=44.008057              PSNR_G=44.217879                PSNR_B=32.441117
MSNR_R=12.974301              MSNR_G=31.953495                MSNR_B=23.570223
MSNR=23.128010
W_PSNR(0.4_0.243_0.357)=39.929646
W_PSNR(0.4_0.3_0.3)=40.600922
W_PSNR(1/3_1/3_1/3)=40.222351
W_MSNR(0.4_0.243_0.357)=21.368989
W_MSNR(0.4_0.3_0.3)=21.846836
W_MSNR(1/3_1/3_1/3)=22.832673
```

```
===============================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
===============================================================================
green_color             1280000                36.849034           13.433123
PSNR_R=43.997031              PSNR_G=44.092170                PSNR_B=32.668515
MSNR_R=-1.850121              MSNR_G=33.580994                MSNR_B=-12.715872
MSNR=17.095519
W_PSNR(0.4_0.243_0.357)=39.975869
W_PSNR(0.4_0.3_0.3)=40.627018
W_PSNR(1/3_1/3_1/3)=40.252572
W_MSNR(0.4_0.243_0.357)=2.880567
W_MSNR(0.4_0.3_0.3)=5.519488
W_MSNR(1/3_1/3_1/3)=6.338334
```

```
===============================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
===============================================================================
red_color               1280000                36.514872           14.507519
PSNR_R=44.036207              PSNR_G=43.992171                PSNR_B=32.291792
MSNR_R=42.897142              MSNR_G=39.006487                MSNR_B=26.229280
MSNR=32.719331
W_PSNR(0.4_0.243_0.357)=39.832750
W_PSNR(0.4_0.3_0.3)=40.499672
W_PSNR(1/3_1/3_1/3)=40.106723
W_MSNR(0.4_0.243_0.357)=36.001286
W_MSNR(0.4_0.3_0.3)=36.729587
W_MSNR(1/3_1/3_1/3)=36.044303
```

Embedding combination 2_4_2

```
===============================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
===============================================================================
peppers_color              1280000                36.104337           15.945815
PSNR_R=44.153184              PSNR_G=31.776004                PSNR_B=44.829000
MSNR_R=39.147218              MSNR_G=24.632853                MSNR_B=32.769434
MSNR=28.505733
W_PSNR(0.4_0.243_0.357)=41.386796
W_PSNR(0.4_0.3_0.3)=40.642775
W_PSNR(1/3_1/3_1/3)=40.252730
W_MSNR(0.4_0.243_0.357)=33.343358
W_MSNR(0.4_0.3_0.3)=32.879573
W_MSNR(1/3_1/3_1/3)=32.183168
```

```
===============================================================================
cover_image          Size_secret_data              PSNR               MSE
                                                    dB
===============================================================================
lena_color              1280000                36.152823           15.768779
PSNR_R=44.138741              PSNR_G=31.831845                PSNR_B=44.803769
MSNR_R=41.119610              MSNR_G=23.619245                MSNR_B=37.142391
MSNR=30.183052
W_PSNR(0.4_0.243_0.357)=41.385580
W_PSNR(0.4_0.3_0.3)=40.646180
W_PSNR(1/3_1/3_1/3)=40.258118
W_MSNR(0.4_0.243_0.357)=35.447154
W_MSNR(0.4_0.3_0.3)=34.676335
W_MSNR(1/3_1/3_1/3)=33.960416
```

```
================================================================================
cover_image          Size_secret_data            PSNR               MSE
                                                  dB
================================================================================
baboon_color              1280000                36.132448          15.842931
PSNR_R=44.147507              PSNR_G=31.807386                    PSNR_B=44.830803
MSNR_R=38.273871              MSNR_G=25.332876                    MSNR_B=37.165546
MSNR=29.492850
W_PSNR(0.4_0.243_0.357)=41.392794
W_PSNR(0.4_0.3_0.3)=40.650460
W_PSNR(1/3_1/3_1/3)=40.261899
W_MSNR(0.4_0.243_0.357)=34.733537
W_MSNR(0.4_0.3_0.3)=34.059075
W_MSNR(1/3_1/3_1/3)=33.590764
```

```
================================================================================
cover_image          Size_secret_data            PSNR               MSE
                                                  dB
================================================================================
barbra_color              1280000                36.098486          15.967310
PSNR_R=44.137269              PSNR_G=31.769880                    PSNR_B=44.840488
MSNR_R=38.587899              MSNR_G=23.821779                    MSNR_B=36.125079
MSNR=28.801455
W_PSNR(0.4_0.243_0.357)=41.383042
W_PSNR(0.4_0.3_0.3)=40.638018
W_PSNR(1/3_1/3_1/3)=40.249212
W_MSNR(0.4_0.243_0.357)=34.120505
W_MSNR(0.4_0.3_0.3)=33.419217
W_MSNR(1/3_1/3_1/3)=32.844919
```

```
================================================================================
cover_image          Size_secret_data            PSNR               MSE
                                                  dB
================================================================================
Balloon_color              1280000                36.161301          15.738027
PSNR_R=44.150850              PSNR_G=31.837414                    PSNR_B=44.866037
MSNR_R=36.700700              MSNR_G=25.611313                    MSNR_B=39.768795
MSNR=29.956436
W_PSNR(0.4_0.243_0.357)=41.414007
W_PSNR(0.4_0.3_0.3)=40.671375
W_PSNR(1/3_1/3_1/3)=40.284767
W_MSNR(0.4_0.243_0.357)=35.101289
W_MSNR(0.4_0.3_0.3)=34.294313
W_MSNR(1/3_1/3_1/3)=34.026936
```

```
================================================================================
cover_image          Size_secret_data            PSNR               MSE
                                                  dB
================================================================================
Blue_color              1280000                36.193658          15.621206
PSNR_R=44.008057              PSNR_G=31.890668                    PSNR_B=44.695241
MSNR_R=12.974301              MSNR_G=19.626284                    MSNR_B=35.824347
MSNR=22.663640
W_PSNR(0.4_0.243_0.357)=41.308856
W_PSNR(0.4_0.3_0.3)=40.578996
W_PSNR(1/3_1/3_1/3)=40.197989
W_MSNR(0.4_0.243_0.357)=22.748199
W_MSNR(0.4_0.3_0.3)=21.824910
W_MSNR(1/3_1/3_1/3)=22.808311
```

```
================================================================================
cover_image          Size_secret_data            PSNR               MSE
                                                  dB
================================================================================
green_color              1280000                36.286560          15.290594
PSNR_R=43.997031              PSNR_G=31.935933                    PSNR_B=45.972875
MSNR_R=-1.850121              MSNR_G=21.424758                    MSNR_B=0.588488
MSNR=16.533045
W_PSNR(0.4_0.243_0.357)=41.771560
W_PSNR(0.4_0.3_0.3)=40.971455
W_PSNR(1/3_1/3_1/3)=40.635280
W_MSNR(0.4_0.243_0.357)=4.676258
W_MSNR(0.4_0.3_0.3)=5.863925
W_MSNR(1/3_1/3_1/3)=6.721042
```

```
================================================================================
cover_image          Size_secret_data           PSNR              MSE
                                                 dB
================================================================================
red_color              1280000               36.092366         15.989827
PSNR_R=44.036207              PSNR_G=31.777991              PSNR_B=44.662545
MSNR_R=42.897142              MSNR_G=26.792307              MSNR_B=38.600032
MSNR=32.296825
W_PSNR(0.4_0.243_0.357)=41.281063
W_PSNR(0.4_0.3_0.3)=40.546643
W_PSNR(1/3_1/3_1/3)=40.158914
W_MSNR(0.4_0.243_0.357)=37.449599
W_MSNR(0.4_0.3_0.3)=36.776558
W_MSNR(1/3_1/3_1/3)=36.096494
```

Embedding combination 4_2_2

```
================================================================================
cover_image          Size_secret_data           PSNR              MSE
                                                 dB
================================================================================
peppers_color              1280000               36.151642         15.773067
PSNR_R=31.829641              PSNR_G=44.132380              PSNR_B=44.829000
MSNR_R=26.823674              MSNR_G=36.989228              MSNR_B=32.769434
MSNR=28.553039
W_PSNR(0.4_0.243_0.357)=39.459978
W_PSNR(0.4_0.3_0.3)=39.420270
W_PSNR(1/3_1/3_1/3)=40.263674
W_MSNR(0.4_0.243_0.357)=31.416540
W_MSNR(0.4_0.3_0.3)=31.657068
W_MSNR(1/3_1/3_1/3)=32.194112
```

```
================================================================================
cover_image          Size_secret_data           PSNR              MSE
                                                 dB
================================================================================
lena_color             1280000               35.954113         16.507035
PSNR_R=31.609777              PSNR_G=44.178162              PSNR_B=44.803769
MSNR_R=28.590647              MSNR_G=35.965563              MSNR_B=37.142391
MSNR=29.984342
W_PSNR(0.4_0.243_0.357)=39.374150
W_PSNR(0.4_0.3_0.3)=39.338490
W_PSNR(1/3_1/3_1/3)=40.197236
W_MSNR(0.4_0.243_0.357)=33.435724
W_MSNR(0.4_0.3_0.3)=33.368645
W_MSNR(1/3_1/3_1/3)=33.899534
```

```
================================================================================
cover_image          Size_secret_data           PSNR              MSE
                                                 dB
================================================================================
baboon_color              1280000               36.173160         15.695108
PSNR_R=31.852268              PSNR_G=44.151892              PSNR_B=44.830803
MSNR_R=25.978631              MSNR_G=37.677382              MSNR_B=37.165546
MSNR=29.533562
W_PSNR(0.4_0.243_0.357)=39.474413
W_PSNR(0.4_0.3_0.3)=39.435716
W_PSNR(1/3_1/3_1/3)=40.278321
W_MSNR(0.4_0.243_0.357)=32.815156
W_MSNR(0.4_0.3_0.3)=32.844331
W_MSNR(1/3_1/3_1/3)=33.607186
```

```
==================================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==================================================================================
barbra_color              1280000                36.163352          15.730596
PSNR_R=31.840527            PSNR_G=44.158293                     PSNR_B=44.840488
MSNR_R=26.291158            MSNR_G=36.210191                     MSNR_B=36.125079
MSNR=28.866321
W_PSNR(0.4_0.243_0.357)=39.474730
W_PSNR(0.4_0.3_0.3)=39.435845
W_PSNR(1/3_1/3_1/3)=40.279769
W_MSNR(0.4_0.243_0.357)=32.212193
W_MSNR(0.4_0.3_0.3)=32.217044
W_MSNR(1/3_1/3_1/3)=32.875476

==================================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==================================================================================
Balloon_color               1280000                36.182630          15.660925
PSNR_R=31.860033            PSNR_G=44.168341                     PSNR_B=44.866037
MSNR_R=24.409883            MSNR_G=37.942241                     MSNR_B=39.768795
MSNR=29.977765
W_PSNR(0.4_0.243_0.357)=39.494095
W_PSNR(0.4_0.3_0.3)=39.454326
W_PSNR(1/3_1/3_1/3)=40.298137
W_MSNR(0.4_0.243_0.357)=33.181378
W_MSNR(0.4_0.3_0.3)=33.077264
W_MSNR(1/3_1/3_1/3)=34.040306

==================================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==================================================================================
Blue_color               1280000               38.275298          9.672763
PSNR_R=34.260334            PSNR_G=44.271536                     PSNR_B=44.695241
MSNR_R=3.226578           MSNR_G=32.007152                     MSNR_B=35.824347
MSNR=24.745281
W_PSNR(0.4_0.243_0.357)=40.418318
W_PSNR(0.4_0.3_0.3)=40.394167
W_PSNR(1/3_1/3_1/3)=41.075704
W_MSNR(0.4_0.243_0.357)=21.857661
W_MSNR(0.4_0.3_0.3)=21.640081
W_MSNR(1/3_1/3_1/3)=23.686026

==================================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==================================================================================
green_color              1280000                34.458233          23.294696
PSNR_R=29.949828            PSNR_G=44.221563                     PSNR_B=45.972875
MSNR_R=-15.897324            MSNR_G=33.710387                     MSNR_B=0.588488
MSNR=14.704718
W_PSNR(0.4_0.243_0.357)=39.138087
W_PSNR(0.4_0.3_0.3)=39.038263
W_PSNR(1/3_1/3_1/3)=40.048089
W_MSNR(0.4_0.243_0.357)=2.042785
W_MSNR(0.4_0.3_0.3)=3.930733
W_MSNR(1/3_1/3_1/3)=6.133851

==================================================================================
cover_image          Size_secret_data              PSNR                MSE
                                                    dB
==================================================================================
red_color              1280000               35.467981          18.462150
PSNR_R=31.090977            PSNR_G=44.008582                     PSNR_B=44.662545
MSNR_R=29.951912            MSNR_G=39.022898                     MSNR_B=38.600032
MSNR=31.672440
W_PSNR(0.4_0.243_0.357)=39.075005
W_PSNR(0.4_0.3_0.3)=39.037729
W_PSNR(1/3_1/3_1/3)=39.920701
W_MSNR(0.4_0.243_0.357)=35.243541
W_MSNR(0.4_0.3_0.3)=35.267644
W_MSNR(1/3_1/3_1/3)=35.858281
```

Appendix E: Adaptive LSB Algorithm (ALSB) for Color Images

E.1 The main program

```
clc
clear
cover_image=imread('C:\Users\hajer\Desktop\thesis\COLOR SCALE\LSB_color\red.jpg');
Red = cover_image(:,:,1);
Green = cover_image(:,:,2);
Blue = cover_image(:,:,3);
[M N L]=size(cover_image);
S=sum(cover_image(:,:,1));
avg_R=sum(S)/(M*N*L);
S2=sum(cover_image(:,:,2));
avg_G=sum(S2)/(M*N*L);
S3=sum(cover_image(:,:,3));
avg_B=sum(S3)/(M*N*L);
disp('=======================================================')
disp('cover_image   Size_secret_data     PSNR        MSE    ')
disp('                  dB          ')
p='red_color';
disp('=======================================================')
secrt_image=imread('C:\Users\hajer\Desktop\thesis\1.jpg');
[m n l]=size(secrt_image);
k=1;
for i=1:m
for j=1:n
str = dec2bin(secrt_image(i,j),8);
for q=1:8
aa=str2num(str(q));
secret_massege(k)=aa;
k=k+1;
end
end
end
[e r]=size(secret_massege);
if (avg_B>=avg_R)&&(avg_B>=avg_G)&&(avg_R>=avg_G)%b=4 r=3 g=1
%blue=4; red=3; green=1;
blue=1; red=3; green=4;
end
if (avg_B>=avg_R)&&(avg_B>=avg_G)&&(avg_G>=avg_R)%b=4 r=1 g=3
%blue=4; red=1; green=3;
blue=1; red=4; green=3;
end
if (avg_G>=avg_R)&&(avg_G>=avg_B)&&(avg_B>=avg_R)%b=3 r=1 g=4
%blue=3; red=1; green=4;
blue=3; red=4; green=1;
end
if (avg_G>=avg_R)&&(avg_G>=avg_B)&&(avg_R>=avg_B)%b=1 r=3 g=4
% blue=1; red=3; green=4;
blue=4; red=3; green=1;
end
```

```
if (avg_R>=avg_B)&&(avg_R>=avg_G)&&(avg_G>=avg_B)%b=1 r=4 g=3
% blue=1; red=4;green=3;
blue=4; red=1;green=3;
end
if (avg_R>=avg_B)&&(avg_R>=avg_G)&&(avg_B>=avg_G)%b=3 r=4 g=1
%blue=3;red=4;green=1;
blue=3;red=1;green=4;
end
[stego_image,y,u]=embedding(red,green,blue,cover_image,secret_massege);
[MSE_R,MSE_G,MSE_B ] = MSE( y,u,cover_image,stego_image);
mse = (MSE_R + MSE_B + MSE_G)/3;
[PSNR,PSNR_R,PSNR_G,PSNR_B ] = PSNR(mse,MSE_R,MSE_G,MSE_B );
disp(sprintf('%s     %d       %f       %f        ' ,p,r,PSNR,mse));
disp(sprintf('avg_R=%f  avg_G=%f  avg_B=%f ',avg_R,avg_G,avg_B));
disp(sprintf('PSNR_R=%f  PSNR_G=%f  PSNR_B=%f ',PSNR_R,PSNR_G,PSNR_B));
[ MSNR,MSNR_R,MSNR_G,MSNR_B ] = MSNR(cover_image,MSE_R,MSE_G,MSE_B,mse);
disp(sprintf('MSNR_R=%f  MSNR_G=%f  MSNR_B=%f',MSNR_R,MSNR_G,MSNR_B));
disp(sprintf('MSNR=%f',MSNR));
Actual_PSNR_weight = A_PSNR( cover_image,MSE_R,MSE_G,MSE_B );
disp(sprintf('Actual_PSNR_weight(1/3_1/3_1/3)=%f',Actual_PSNR_weight));
disp(sprintf('W_PSNR(0.4_0.243_0.357)=%f',(0.4*PSNR_R+0.243*PSNR_G+0.357*PSNR_B)));
disp(sprintf('W_PSNR(0.4_0.3_0.3)=%f',(0.4*PSNR_R+0.3*PSNR_G+0.3*PSNR_B)));
disp(sprintf('W_PSNR(1/3_1/3_1/3)=%f',(1/3*PSNR_R+1/3*PSNR_G+1/3*PSNR_B)));
disp(sprintf('W_MSNR(0.4_0.243_0.357)=%f',(0.4*MSNR_R+0.243*MSNR_G+0.357*MSNR_B)));
disp(sprintf('W_MSNR(0.4_0.3_0.3)=%f',(0.4*MSNR_R+0.3*MSNR_G+0.3*MSNR_B)));
disp(sprintf('W_MSNR(1/3_1/3_1/3)=%f',(1/3*MSNR_R+1/3*MSNR_G+1/3*MSNR_B)));
[var_orgR, var_noiseR] = snr(stego_image(:,:,1),cover_image(:,:,1));
[var_orgG, var_noiseG] = snr(stego_image(:,:,2),cover_image(:,:,2));
[var_orgB, var_noiseB] = snr(stego_image(:,:,3),cover_image(:,:,3));
SNR= 10*log10((var_orgR + var_orgG + var_orgB)/(var_noiseR + var_noiseG + var_noiseB));
disp(sprintf('SNR=%f',SNR));
disp('==========================================================')
figure
subplot(2,2,[1,3]);
imshow(cover_image);
title('Cover image')
subplot(2,2,[2,4]);
imshow(stego_image);
title('Stego image')
```

E.2 Embedding function

```
function [ stego_image,y,u]=embedding(red,green,blue,cover_image,secret_massege)
u=0;
y=0;
[e r]=size(secret_massege);
[M N L]=size(cover_image);
stego_image=cover_image;
k=1;
```

```
stego_image(1,1,3)=cover_image(1,1,3)-(mod(cover_image(1,1,3),2^3))+blue;
stego_image(1,1,2)=cover_image(1,1,2)-(mod(cover_image(1,1,2),2^3))+green;
stego_image(1,1,1)=cover_image(1,1,1)-(mod(cover_image(1,1,1),2^3))+red;
for j=2:N
if (k<=r)
EC=blue;
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
stego_image(1,j,3)=cover_image(1,j,3)-(mod(cover_image(1,j,3),2^EC))+DEC;
EC=green;
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
stego_image(1,j,2)=cover_image(1,j,2)-(mod(cover_image(1,j,2),2^EC))+DEC;
EC=red;
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
stego_image(1,j,1)=cover_image(1,j,1)-(mod(cover_image(1,j,1),2^EC))+DEC;
end
end
for i=2:M
for j=1:N
if (k<=r)
EC=blue;
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
```

```
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
stego_image(i,j,3)=cover_image(i,j,3)-(mod(cover_image(i,j,3),2^EC))+DEC;
EC=green;
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
stego_image(i,j,2)=cover_image(i,j,2)-(mod(cover_image(i,j,2),2^EC))+DEC;
EC=red;
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=secret_massege(k:k+EC-1-q);
k=k+EC;
else
s=secret_massege(k:k+EC-1);
k=k+EC;
end
a = num2str(s);
DEC= bin2dec(a);
stego_image(i,j,1)=cover_image(i,j,1)-(mod(cover_image(i,j,1),2^EC))+DEC;
y= i;
u=j;
end
end
end
end
```

E3 Extraction function

```
function [ output ] = extraction( r,stego_image )
[M N L]=size(stego_image);
k=1;
kk=1;
blue=mod(stego_image(1,1,3),2^3);
green=mod(stego_image(1,1,2),2^3);
red=mod(stego_image(1,1,1),2^3);
for j=2:N
if (kk<=r)
EC=blue;
RES=mod(stego_image(1,j,3),2^EC);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
```

```
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
EC=green;
RES=mod(stego_image(1,j,2),2^EC);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
EC=red;
RES=mod(stego_image(1,j,1),2^EC);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
end
```

```
end
for i=2:M
for j=1:N
if (kk<=r)
EC=blue;
RES=mod(stego_image(i,j,3),2^EC);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
EC=green;
RES=mod(stego_image(i,j,2),2^EC);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
EC=red;
RES=mod(stego_image(i,j,1),2^EC);
if (k+EC-1)>=r
q=(k+EC-1)-r;
s=dec2bin(RES,EC-q);
for g=1:EC-q
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
```

```
else
s=dec2bin(RES,EC);
for g=1:EC
a=str2num(s(g));
output(kk)=a;
kk=kk+1;
end
k=k+EC;
end
end
end
end
end
```

Other functions are the same as shown in Appendices D.4, D.5, D.6, D.7 and D.8.

E. 4 Screenshots of Adaptive LSB for color images results for different embedding combinations for cover images with size 512×512

Results for ALSBmin

```
==========================================================================
cover_image         Size_secret_data          PSNR              MSE
                                               dB
===========================================================================
peppers_color                1280000             36.355709         15.049064
avg_R=47.766188                avg_G=37.347483                avg_B=21.205114
PSNR_R=51.132833                PSNR_G=37.932710                 PSNR_B=32.793140
MSNR_R=46.126866                MSNR_G=30.789558                 MSNR_B=20.733574
MSNR=28.757105
W_PSNR(0.4_0.243_0.357)=41.377933
W_PSNR(0.4_0.3_0.3)=41.670888
W_PSNR(1/3_1/3_1/3)=40.619561
W_MSNR(0.4_0.243_0.357)=33.334495
W_MSNR(0.4_0.3_0.3)=33.907686
W_MSNR(1/3_1/3_1/3)=32.549999
```

```
==========================================================================
cover_image         Size_secret_data          PSNR              MSE
                                               dB
===========================================================================
lena_color                  1280000             35.599405         17.911826
avg_R=60.043004                avg_G=33.020903                avg_B=35.184392
PSNR_R=51.143248                PSNR_G=31.651164                PSNR_B=38.697698
MSNR_R=48.124118                MSNR_G=23.438564                MSNR_B=31.036320
MSNR=29.629634
W_PSNR(0.4_0.243_0.357)=41.963610
W_PSNR(0.4_0.3_0.3)=41.561958
W_PSNR(1/3_1/3_1/3)=40.497370
W_MSNR(0.4_0.243_0.357)=36.025185
W_MSNR(0.4_0.3_0.3)=35.592113
W_MSNR(1/3_1/3_1/3)=34.199667
```

```
=============================================================================
cover_image          Size_secret_data            PSNR              MSE
                                                   dB
=============================================================================
baboon_color              1280000              36.326615          15.150216
avg_R=43.225210              avg_G=40.336053              avg_B=35.168683
PSNR_R=51.135977              PSNR_G=37.904887              PSNR_B=32.763187
MSNR_R=45.262340              MSNR_G=31.430377              MSNR_B=25.097931
MSNR=29.687017
W_PSNR(0.4_0.243_0.357)=41.361736
W_PSNR(0.4_0.3_0.3)=41.654813
W_PSNR(1/3_1/3_1/3)=40.601350
W_MSNR(0.4_0.243_0.357)=34.702479
W_MSNR(0.4_0.3_0.3)=35.063428
W_MSNR(1/3_1/3_1/3)=33.930216
```

```
=============================================================================
cover_image          Size_secret_data            PSNR              MSE
                                                   dB
=============================================================================
barbra_color              1280000              36.307405          15.217378
avg_R=44.869422              avg_G=34.041904              avg_B=31.163663
PSNR_R=51.137549              PSNR_G=37.915266              PSNR_B=32.734660
MSNR_R=45.588180              MSNR_G=29.967164              MSNR_B=24.019252
MSNR=29.010375
W_PSNR(0.4_0.243_0.357)=41.354703
W_PSNR(0.4_0.3_0.3)=41.649998
W_PSNR(1/3_1/3_1/3)=40.595825
W_MSNR(0.4_0.243_0.357)=34.092166
W_MSNR(0.4_0.3_0.3)=34.431197
W_MSNR(1/3_1/3_1/3)=33.191532
```

```
=============================================================================
cover_image          Size_secret_data            PSNR              MSE
                                                   dB
=============================================================================
Balloon_color             1280000              35.629944          17.786314
avg_R=36.050513              avg_G=41.506289              avg_B=47.266870
PSNR_R=31.849739              PSNR_G=37.964555              PSNR_B=51.172896
MSNR_R=24.399590              MSNR_G=31.738455              MSNR_B=46.075654
MSNR=29.425080
W_PSNR(0.4_0.243_0.357)=40.234006
W_PSNR(0.4_0.3_0.3)=39.481131
W_PSNR(1/3_1/3_1/3)=40.329063
W_MSNR(0.4_0.243_0.357)=33.921289
W_MSNR(0.4_0.3_0.3)=33.104068
W_MSNR(1/3_1/3_1/3)=34.071233
```

```
=============================================================================
cover_image          Size_secret_data            PSNR              MSE
                                                   dB
=============================================================================
Blue_color                1280000              37.441416          11.720300
avg_R=2.386333              avg_G=20.710935              avg_B=30.610770
PSNR_R=34.264084              PSNR_G=38.004599              PSNR_B=51.086430
MSNR_R=3.230328              MSNR_G=25.740214              MSNR_B=42.215536
MSNR=23.911399
W_PSNR(0.4_0.243_0.357)=41.178606
W_PSNR(0.4_0.3_0.3)=40.432942
W_PSNR(1/3_1/3_1/3)=41.118371
W_MSNR(0.4_0.243_0.357)=22.617950
W_MSNR(0.4_0.3_0.3)=21.678856
W_MSNR(1/3_1/3_1/3)=23.728693
```

```
===========================================================================
cover_image         Size_secret_data          PSNR              MSE
                                               dB
===========================================================================
green_color             1280000               34.235510          24.520503
avg_R=0.433572              avg_G=25.343124              avg_B=0.457298
PSNR_R=29.949523              PSNR_G=51.100713              PSNR_B=39.514467
MSNR_R=-15.897628              MSNR_G=40.589537              MSNR_B=-5.869920
MSNR=14.481995
W_PSNR(0.4_0.243_0.357)=38.503947
W_PSNR(0.4_0.3_0.3)=39.164363
W_PSNR(1/3_1/3_1/3)=40.188234
W_MSNR(0.4_0.243_0.357)=1.408645
W_MSNR(0.4_0.3_0.3)=4.056834
W_MSNR(1/3_1/3_1/3)=6.273997
```

```
===========================================================================
cover_image         Size_secret_data          PSNR              MSE
                                               dB
===========================================================================
red_color             1280000               35.911542          16.669640
avg_R=74.553097              avg_G=47.877857              avg_B=42.295417
PSNR_R=51.149553              PSNR_G=37.661906              PSNR_B=32.290911
MSNR_R=50.010489              MSNR_G=32.676221              MSNR_B=26.228399
MSNR=32.116000
W_PSNR(0.4_0.243_0.357)=41.139520
W_PSNR(0.4_0.3_0.3)=41.445666
W_PSNR(1/3_1/3_1/3)=40.367457
W_MSNR(0.4_0.243_0.357)=37.308056
W_MSNR(0.4_0.3_0.3)=37.675582
W_MSNR(1/3_1/3_1/3)=36.305036
```

Results of ALSBMax

```
===========================================================================
cover_image         Size_secret_data          PSNR              MSE
                                               dB
===========================================================================
peppers_color             1280000               35.618598          17.832841
avg_R=47.766188              avg_G=37.347483              avg_B=21.205114
PSNR_R=31.808135              PSNR_G=38.078363              PSNR_B=51.177328
MSNR_R=26.802168              MSNR_G=30.935212              MSNR_B=39.117761
MSNR=28.019995
W_PSNR(0.4_0.243_0.357)=40.246602
W_PSNR(0.4_0.3_0.3)=39.499961
W_PSNR(1/3_1/3_1/3)=40.354609
W_MSNR(0.4_0.243_0.357)=32.203164
W_MSNR(0.4_0.3_0.3)=31.736759
W_MSNR(1/3_1/3_1/3)=32.285047
```

```
===========================================================================
cover_image         Size_secret_data          PSNR              MSE
                                               dB
===========================================================================
lena_color             1280000               35.558127          18.082883
avg_R=60.043004              avg_G=33.020903              avg_B=35.184392
PSNR_R=31.601155              PSNR_G=51.158319              PSNR_B=38.697698
MSNR_R=28.582025              MSNR_G=42.945720              MSNR_B=31.036320
MSNR=29.588356
W_PSNR(0.4_0.243_0.357)=38.887012
W_PSNR(0.4_0.3_0.3)=39.597267
W_PSNR(1/3_1/3_1/3)=40.485724
W_MSNR(0.4_0.243_0.357)=32.948586
W_MSNR(0.4_0.3_0.3)=33.627422
W_MSNR(1/3_1/3_1/3)=34.188022
```

```
==============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
==============================================================================
baboon_color               1280000               35.632098          17.777493
avg_R=43.225210               avg_G=40.336053               avg_B=35.168683
PSNR_R=31.847124               PSNR_G=37.987395               PSNR_B=51.151457
MSNR_R=25.973488               MSNR_G=31.512885               MSNR_B=43.486201
MSNR=28.992500
W_PSNR(0.4_0.243_0.357)=40.230857
W_PSNR(0.4_0.3_0.3)=39.480505
W_PSNR(1/3_1/3_1/3)=40.328659
W_MSNR(0.4_0.243_0.357)=33.571600
W_MSNR(0.4_0.3_0.3)=32.889121
W_MSNR(1/3_1/3_1/3)=33.657524
```

```
==============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
==============================================================================
barbra_color               1280000               35.640277          17.744047
avg_R=44.869422               avg_G=34.041904               avg_B=31.163663
PSNR_R=31.852410               PSNR_G=38.007978               PSNR_B=51.149499
MSNR_R=26.303041               MSNR_G=30.059876               MSNR_B=42.434090
MSNR=28.343246
W_PSNR(0.4_0.243_0.357)=40.237274
W_PSNR(0.4_0.3_0.3)=39.488207
W_PSNR(1/3_1/3_1/3)=40.336629
W_MSNR(0.4_0.243_0.357)=32.974737
W_MSNR(0.4_0.3_0.3)=32.269406
W_MSNR(1/3_1/3_1/3)=32.932336
```

```
==============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
==============================================================================
Balloon_color               1280000               36.383310          14.953725
avg_R=36.050513               avg_G=41.506289               avg_B=47.266870
PSNR_R=51.158755               PSNR_G=37.879918               PSNR_B=32.845685
MSNR_R=43.708606               MSNR_G=31.653818               MSNR_B=27.748443
MSNR=30.178445
W_PSNR(0.4_0.243_0.357)=41.394232
W_PSNR(0.4_0.3_0.3)=41.681183
W_PSNR(1/3_1/3_1/3)=40.628120
W_MSNR(0.4_0.243_0.357)=35.081514
W_MSNR(0.4_0.3_0.3)=35.304121
W_MSNR(1/3_1/3_1/3)=34.370289
```

```
==============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
==============================================================================
blue_color               1280000               36.084702          16.018071
avg_R=2.386333               avg_G=20.710935               avg_B=30.610770
PSNR_R=51.122496               PSNR_G=37.923412               PSNR_B=32.441405
MSNR_R=20.088740               MSNR_G=25.659028               MSNR_B=23.570511
MSNR=22.554684
W_PSNR(0.4_0.243_0.357)=41.245969
W_PSNR(0.4_0.3_0.3)=41.558443
W_PSNR(1/3_1/3_1/3)=40.495771
W_MSNR(0.4_0.243_0.357)=22.685312
W_MSNR(0.4_0.3_0.3)=22.804358
W_MSNR(1/3_1/3_1/3)=23.106093
```

```
==============================================================================
cover_image          Size_secret_data              PSNR              MSE
                                                    dB
==============================================================================
green_color               1280000               35.818776          17.029533
avg_R=0.433572               avg_G=25.343124               avg_B=0.457298
PSNR_R=51.140316               PSNR_G=31.764235               PSNR_B=39.51446
MSNR_R=5.293165               MSNR_G=21.253060               MSNR_B=-5.869920
MSNR=16.065261
W_PSNR(0.4_0.243_0.357)=42.281501
W_PSNR(0.4_0.3_0.3)=41.839737
W_PSNR(1/3_1/3_1/3)=40.806340
W_MSNR(0.4_0.243_0.357)=5.186198
W_MSNR(0.4_0.3_0.3)=6.732208
W_MSNR(1/3_1/3_1/3)=6.892102
```

```
===============================================================================
cover_image          Size_secret_data          PSNR              MSE
                                                dB
===============================================================================
red_color             1280000               34.976671         20.673467
avg_R=74.553097             avg_G=47.877857               avg_B=42.295417
PSNR_R=31.085299             PSNR_G=37.772163               PSNR_B=51.030292
MSNR_R=29.946234             MSNR_G=32.786478               MSNR_B=44.967780
MSNR=31.181130
W_PSNR(0.4_0.243_0.357)=39.830569
W_PSNR(0.4_0.3_0.3)=39.074856
W_PSNR(1/3_1/3_1/3)=39.962585
W_MSNR(0.4_0.243_0.357)=35.999105
W_MSNR(0.4_0.3_0.3)=35.304771
W_MSNR(1/3_1/3_1/3)=35.900164
```

Appendix F: ATD Algorithm for Color Images

F.1 The main program

```
clc
clear
cover_image=imread('C:\Users\hajer\Desktop\thesis\COLOR SCALE\ATD color\peppers_color (512 x 512).jpg');
disp('=======================================================')
disp('cover_image    Size_secret_data     PSNR       MSE  ')
disp('                              dB         ')
disp('=======================================================')
p='peppers_color.jpg';
secrt_image=imread('C:\Users\hajer\Desktop\thesis\1.jpg');
[m n l]=size(secrt_image);
k=1;
for i=1:m
for j=1:n
str = dec2bin(secrt_image(i,j),8);
for q=1:8
aa=str2num(str(q));
secret_massege(k)=aa;
k=k+1;
end
end
end
k=k-1;
ternary_number=convertto__ternary(secret_massege);
[q size_secret_massege]=size(ternary_number);
[stego_image,y,u]=Embbedding(cover_image,size_secret_massege,ternary_number);
[MSE_R,MSE_G,MSE_B ] = MSE( y,u,cover_image,stego_image);
mse = (MSE_R + MSE_B + MSE_G)/3;
[PSNR,PSNR_R,PSNR_G,PSNR_B ] = PSNR(mse,MSE_R,MSE_G,MSE_B );
disp(sprintf('%s   %d      %f          %2f' ,p,k,PSNR,mse));
disp(sprintf('PSNR_R=%f  PSNR_G=%f PSNR_B=%f ',PSNR_R,PSNR_G,PSNR_B));
[ MSNR,MSNR_R,MSNR_G,MSNR_B ] = MSNR(cover_image,MSE_R,MSE_G,MSE_B,mse);
disp(sprintf('MSNR_R=%f  MSNR_G=%f  MSNR_B=%f ',MSNR_R,MSNR_G,MSNR_B));
disp(sprintf('MSNR=%f',MSNR));
disp(sprintf('W_PSNR(0.4_0.243_0.357)=%f',(0.4*PSNR_R+0.243*PSNR_G+0.357*PSNR_B)));
disp(sprintf('W_PSNR(0.4_0.3_0.3)=%f',(0.4*PSNR_R+0.3*PSNR_G+0.3*PSNR_B)));
disp(sprintf('W_PSNR(1/3_1/3_1/3)=%f ',(1/3*PSNR_R+1/3*PSNR_G+1/3*PSNR_B)));
disp(sprintf('W_MSNR(0.4_0.243_0.357)=%f',(0.4*MSNR_R+0.243*MSNR_G+0.357*MSNR_B)));
disp(sprintf('W_MSNR(0.4_0.3_0.3)=%f',(0.4*MSNR_R+0.3*MSNR_G+0.3*MSNR_B)));
disp(sprintf('W_MSNR(1/3_1/3_1/3)=%f ',(1/3*MSNR_R+1/3*MSNR_G+1/3*MSNR_B)));
Actual_PSNR_weight = A_PSNR( cover_image,MSE_R,MSE_G,MSE_B );
disp(sprintf('Actual_PSNR_weight(1/3_1/3_1/3)=%f ',Actual_PSNR_weight));
[var_orgR, var_noiseR] = snr(stego_image(:,:,1),cover_image(:,:,1));
[var_orgG, var_noiseG] = snr(stego_image(:,:,2),cover_image(:,:,2));
[var_orgB, var_noiseB] = snr(stego_image(:,:,3),cover_image(:,:,3));
SNR= 10*log10((var_orgR + var_orgG + var_orgB)/(var_noiseR + var_noiseG + var_noiseB));
```

```
disp(sprintf('SNR=%f',SNR));
disp('=======================================================')
figure
subplot(2,2,[1,3]);
imshow(cover_image);
title('Cover image')
subplot(2,2,[2,4]);
imshow(stego_image);
title('Stego image')
output = extraction( stego_image,size_secret_massege);
[nn mm]=size(output);
for i=1:mm
e(i)=output(i)-ternary_number(i);
end
output2=convert_to_Binary(output);
[nn mm]=size(secret_massege);
```

F.2 Embedding function

```
function [ stego_image,y,u ]  = Embbedding( cover_image,size_secret_massege,ternary_number )
stego_image=cover_image;
k=1;
[M N L]=size(cover_image);
y=0;
u=0;
for i=1:M
for j=1:N
cover_pixel_binary = dec2bin(cover_image(i,j,3),8);
for ii=1:6
sub1(ii)=cover_pixel_binary(ii);
end
sub1_dec=bin2dec(sub1);
sub2=cover_pixel_binary(7:8);
sub2_dec=bin2dec(sub2);
if (sub1_dec==63)
sub1_dec=62;
end
if (sub1_dec==0)
sub1_dec=1;
end
if (sub2_dec==3)
sub2_dec=2;
end
if (sub2_dec==0)
sub2_dec=1;
end
if(k<=size_secret_massege)
if (mod(sub1_dec,3)==ternary_number(k))
sub1_stego=sub1_dec;
elseif(mod(sub1_dec+1,3)==ternary_number(k))
sub1_stego=sub1_dec+1;
```

```
else
sub1_stego=sub1_dec-1;
end
end
k=k+1;
if(k<=size_secret_massege)
v=sub1_stego*4+sub2_dec;
if (mod(v,3)==ternary_number(k))
stego_image(i,j,3)=v;
elseif(mod(v+1,3)==ternary_number(k))
stego_image(i,j,3)=v+1;
else
stego_image(i,j,3)=v-1;
end
k=k+1;
y= i;
u=j ;
end
cover_pixel_binary = dec2bin(cover_image(i,j,2),8);
for ii=1:6
sub1(ii)=cover_pixel_binary(ii);
end
sub1_dec=bin2dec(sub1);
sub2=cover_pixel_binary(7:8);
sub2_dec=bin2dec(sub2);
if (sub1_dec==63)
sub1_dec=62;
end
if (sub1_dec==0)
sub1_dec=1;
end
if (sub2_dec==3)
sub2_dec=2;
end
if (sub2_dec==0)
sub2_dec=1;
end
if(k<=size_secret_massege)
if (mod(sub1_dec,3)==ternary_number(k))
sub1_stego=sub1_dec;
elseif(mod(sub1_dec+1,3)==ternary_number(k))
sub1_stego=sub1_dec+1;
else
sub1_stego=sub1_dec-1;
end
end
k=k+1;
if(k<=size_secret_massege)
v=sub1_stego*4+sub2_dec;
if (mod(v,3)==ternary_number(k))
stego_image(i,j,2)=v;
elseif(mod(v+1,3)==ternary_number(k))
```

```
stego_image(i,j,2)=v+1;
else
stego_image(i,j,2)=v-1;
end
k=k+1;
y= i;
u=j ;
end
if(k<=size_secret_massege)
cover_pixel_binary = dec2bin(cover_image(i,j,1),8);
for ii=1:6
sub1(ii)=cover_pixel_binary(ii);
end
sub1_dec=bin2dec(sub1);
sub2=cover_pixel_binary(7:8);
sub2_dec=bin2dec(sub2);
if (sub1_dec==63)
sub1_dec=62;
end
if (sub1_dec==0)
sub1_dec=1;
end
if (sub2_dec==3)
sub2_dec=2;
end
if (sub2_dec==0)
sub2_dec=1;
end
if(k<=size_secret_massege)
if (mod(sub1_dec,3)==ternary_number(k))
sub1_stego=sub1_dec;
elseif(mod(sub1_dec+1,3)==ternary_number(k))
sub1_stego=sub1_dec+1;
else
sub1_stego=sub1_dec-1;
end
end
k=k+1;
stego_image(i,j,1)=sub1_stego*4+sub2_dec;
y= i;
u=j ;
else
stego_image(i,j,1)= cover_image(i,j,1);
end
end
 end
```

F.3 Extraction function

```
function [ output ] = extraction( stego_image,size_secret_image)
k=1;
q=1;
```

```
for i=1:512
for j=1:512
if (k<=size_secret_image)
cover_pixel_binary = dec2bin(stego_image(i,j,3),8);
sub1=cover_pixel_binary(1:6);
sub1_dec=bin2dec(sub1);
sub2=cover_pixel_binary(7:8);
sub2_dec=bin2dec(sub2);
output(k)=mod(sub1_dec,3);
k=k+1;
q=q+1;
output(k)=mod(stego_image(i,j,3),3);
k=k+1;
q=q+1;
else
continue
end
if (k<=size_secret_image)
cover_pixel_binary = dec2bin(stego_image(i,j,2),8);
sub1=cover_pixel_binary(1:6);
sub1_dec=bin2dec(sub1);
sub2=cover_pixel_binary(7:8);
sub2_dec=bin2dec(sub2);
output(k)=mod(sub1_dec,3);
k=k+1;
q=q+1;
else
continue
end
if (k<=size_secret_image)
cover_pixel_binary = dec2bin(stego_image(i,j,1),8);
sub1=cover_pixel_binary(1:6);
sub1_dec=bin2dec(sub1);
sub2=cover_pixel_binary(7:8);
sub2_dec=bin2dec(sub2);
output(k)=mod(sub1_dec,3);
k=k+1;
q=q+1;
else
continue
end
  end
  end
  end
```

Other functions are the same as for ATD for gray scale images given in Appendix A.

F.4 Screenshots of ATD for color images results for different embedding ternary digits combinations with 8 BPP for cover images with size 512×512

Embedding combination 1_1_2

```
===========================================================================
cover_image         Size_secret_data          PSNR               MSE
                                               dB
===========================================================================
peppers_color.jpg             1280000          37.417430          11.785210
PSNR_R=37.653454              PSNR_G=37.495215                PSNR_B=37.120967
MSNR_R=32.647487              MSNR_G=30.352063                MSNR_B=25.061401
MSNR=29.818827
W_PSNR(0.4_0.243_0.357)=37.424904
W_PSNR(0.4_0.3_0.3)=37.446236
W_PSNR(1/3_1/3_1/3)=37.423212
W_MSNR(0.4_0.243_0.357)=29.381466
W_MSNR(0.4_0.3_0.3)=29.683034
W_MSNR(1/3_1/3_1/3)=29.353650
Actual_PSNR_weight(1/3_1/3_1/3)=36.730876
===========================================================================
```

```
===========================================================================
cover_image         Size_secret_data          PSNR               MSE
                                               dB
===========================================================================
lena_color.jpg             1280000          37.548064          11.435995
PSNR_R=37.610816              PSNR_G=37.643593                PSNR_B=37.394040
MSNR_R=34.591686              MSNR_G=29.430994                MSNR_B=29.732663
MSNR=31.578293
W_PSNR(0.4_0.243_0.357)=37.541392
W_PSNR(0.4_0.3_0.3)=37.555617
W_PSNR(1/3_1/3_1/3)=37.549483
W_MSNR(0.4_0.243_0.357)=31.602967
W_MSNR(0.4_0.3_0.3)=31.585771
W_MSNR(1/3_1/3_1/3)=31.251781
Actual_PSNR_weight(1/3_1/3_1/3)=36.912974
===========================================================================
```

```
===========================================================================
cover_image         Size_secret_data          PSNR               MSE
                                               dB
===========================================================================
barbra_color.jpg             1280000          37.565833          11.389301
PSNR_R=37.644377              PSNR_G=37.662399                PSNR_B=37.395864
MSNR_R=32.095008              MSNR_G=29.714297                MSNR_B=28.680456
MSNR=30.268802
W_PSNR(0.4_0.243_0.357)=37.560037
W_PSNR(0.4_0.3_0.3)=37.575230
W_PSNR(1/3_1/3_1/3)=37.567547
W_MSNR(0.4_0.243_0.357)=30.297500
W_MSNR(0.4_0.3_0.3)=30.356429
W_MSNR(1/3_1/3_1/3)=30.163254
Actual_PSNR_weight(1/3_1/3_1/3)=37.007569
===========================================================================
```

```
===========================================================================
cover_image         Size_secret_data          PSNR               MSE
                                               dB
===========================================================================
baboon_color.jpg             1280000          37.560003          11.404600
PSNR_R=37.655576              PSNR_G=37.658859                PSNR_B=37.371869
MSNR_R=31.781940              MSNR_G=31.184348                MSNR_B=29.706613
MSNR=30.920405
W_PSNR(0.4_0.243_0.357)=37.555090
W_PSNR(0.4_0.3_0.3)=37.571449
W_PSNR(1/3_1/3_1/3)=37.562101
W_MSNR(0.4_0.243_0.357)=30.895833
W_MSNR(0.4_0.3_0.3)=30.980064
W_MSNR(1/3_1/3_1/3)=30.890967
Actual_PSNR_weight(1/3_1/3_1/3)=36.919372
===========================================================================
```

```
==========================================================================================
cover_image          Size_secret_data              PSNR                   MSE
                                                    dB
==========================================================================================
Balloon_color.jpg                 1280000          37.425772              11.762595
PSNR_R=37.530423                  PSNR_G=37.606875                  PSNR_B=37.153696
MSNR_R=30.080274                  MSNR_G=31.380775                  MSNR_B=32.056454
MSNR=31.220908
W_PSNR(0.4_0.243_0.357)=37.414510
W_PSNR(0.4_0.3_0.3)=37.440341
W_PSNR(1/3_1/3_1/3)=37.430332
W_MSNR(0.4_0.243_0.357)=31.101792
W_MSNR(0.4_0.3_0.3)=31.063278
W_MSNR(1/3_1/3_1/3)=31.172501
Actual_PSNR_weight(1/3_1/3_1/3)=37.430332
==========================================================================================
```

```
==========================================================================================
cover_image          Size_secret_data              PSNR                   MSE
                                                    dB
==========================================================================================
Blue_color.jpg                 1280000          37.537336              11.464280
PSNR_R=37.556763                  PSNR_G=37.662996                  PSNR_B=37.396405
MSNR_R=6.523007                MSNR_G=25.398611                 MSNR_B=28.525511
MSNR=24.007318
W_PSNR(0.4_0.243_0.357)=37.525330
W_PSNR(0.4_0.3_0.3)=37.540525
W_PSNR(1/3_1/3_1/3)=37.538721
W_MSNR(0.4_0.243_0.357)=18.964673
W_MSNR(0.4_0.3_0.3)=18.786440
W_MSNR(1/3_1/3_1/3)=20.149043
Actual_PSNR_weight(1/3_1/3_1/3)=28.499369
==========================================================================================
```

```
==========================================================================================
cover_image          Size_secret_data              PSNR                   MSE
                                                    dB
==========================================================================================
green_color.jpg                1280000          34.497700              23.083964
PSNR_R=33.537560                PSNR_G=37.649599                 PSNR_B=33.464399
MSNR_R=-12.309591                MSNR_G=27.138423                 MSNR_B=-11.919988
MSNR=14.744185
W_PSNR(0.4_0.243_0.357)=34.510667
W_PSNR(0.4_0.3_0.3)=34.749223
W_PSNR(1/3_1/3_1/3)=34.883853
W_MSNR(0.4_0.243_0.357)=-2.584635
W_MSNR(0.4_0.3_0.3)=-0.358306
W_MSNR(1/3_1/3_1/3)=0.969615
Actual_PSNR_weight(1/3_1/3_1/3)=22.439665
==========================================================================================
```

```
==========================================================================================
cover_image          Size_secret_data              PSNR                   MSE
                                                    dB
==========================================================================================
red_color.jpg                1280000          36.767455              13.687839
PSNR_R=36.336424                PSNR_G=37.066879                 PSNR_B=36.934434
MSNR_R=35.197359                MSNR_G=32.081195                 MSNR_B=30.871922
MSNR=32.971914
W_PSNR(0.4_0.243_0.357)=36.727414
W_PSNR(0.4_0.3_0.3)=36.734964
W_PSNR(1/3_1/3_1/3)=36.779246
W_MSNR(0.4_0.243_0.357)=32.895950
W_MSNR(0.4_0.3_0.3)=32.964879
W_MSNR(1/3_1/3_1/3)=32.716825
Actual_PSNR_weight(1/3_1/3_1/3)=36.779246
==========================================================================================
```

Embedding combination 1_2_1

```
==========================================================================
cover_image          Size_secret_data          PSNR              MSE
                                                dB
==========================================================================
peppers_color.jpg          1280000          37.415784          11.789680
PSNR_R=37.653454           PSNR_G=37.240059                PSNR_B=37.364136
MSNR_R=32.647487           MSNR_G=30.096907                MSNR_B=25.304570
MSNR=29.817180
W_PSNR(0.4_0.243_0.357)=37.449712
W_PSNR(0.4_0.3_0.3)=37.442640
W_PSNR(1/3_1/3_1/3)=37.419216
W_MSNR(0.4_0.243_0.357)=29.406275
W_MSNR(0.4_0.3_0.3)=29.679438
W_MSNR(1/3_1/3_1/3)=29.349655
Actual_PSNR_weight(1/3_1/3_1/3)=36.726881
==========================================================================
```

```
==========================================================================
cover_image          Size_secret_data          PSNR              MSE
                                                dB
==========================================================================
lena_color.jpg          1280000          37.553098          11.422746
PSNR_R=37.610816           PSNR_G=37.400283                PSNR_B=37.652425
MSNR_R=34.591686           MSNR_G=29.187683                MSNR_B=29.991048
MSNR=31.583327
W_PSNR(0.4_0.243_0.357)=37.574511
W_PSNR(0.4_0.3_0.3)=37.560139
W_PSNR(1/3_1/3_1/3)=37.554508
W_MSNR(0.4_0.243_0.357)=31.636086
W_MSNR(0.4_0.3_0.3)=31.590294
W_MSNR(1/3_1/3_1/3)=31.256806
Actual_PSNR_weight(1/3_1/3_1/3)=36.917999
==========================================================================
```

```
==========================================================================
cover_image          Size_secret_data          PSNR              MSE
                                                dB
==========================================================================
barbra_color.jpg          1280000          37.559048          11.407109
PSNR_R=37.644377           PSNR_G=37.385857                PSNR_B=37.652233
MSNR_R=32.095008           MSNR_G=29.437756                MSNR_B=28.936825
MSNR=30.262017
W_PSNR(0.4_0.243_0.357)=37.584362
W_PSNR(0.4_0.3_0.3)=37.569178
W_PSNR(1/3_1/3_1/3)=37.560823
W_MSNR(0.4_0.243_0.357)=30.321825
W_MSNR(0.4_0.3_0.3)=30.350378
W_MSNR(1/3_1/3_1/3)=30.156530
Actual_PSNR_weight(1/3_1/3_1/3)=37.000845
==========================================================================
```

```
==========================================================================
cover_image          Size_secret_data          PSNR              MSE
                                                dB
==========================================================================
baboon_color.jpg          1280000          37.563321          11.395891
PSNR_R=37.655576           PSNR_G=37.405493                PSNR_B=37.633330
MSNR_R=31.781940           MSNR_G=30.930982                MSNR_B=29.968074
MSNR=30.923723
W_PSNR(0.4_0.243_0.357)=37.586864
W_PSNR(0.4_0.3_0.3)=37.573877
W_PSNR(1/3_1/3_1/3)=37.564800
W_MSNR(0.4_0.243_0.357)=30.927607
W_MSNR(0.4_0.3_0.3)=30.982493
W_MSNR(1/3_1/3_1/3)=30.893665
Actual_PSNR_weight(1/3_1/3_1/3)=36.922071
==========================================================================
```

```
==========================================================================
cover_image          Size_secret_data          PSNR              MSE
                                                dB
==========================================================================
Balloon_color.jpg          1280000          37.429199          11.753317
PSNR_R=37.530423           PSNR_G=37.361126                PSNR_B=37.397868
MSNR_R=30.080274           MSNR_G=31.135025                MSNR_B=32.300626
MSNR=31.224335
W_PSNR(0.4_0.243_0.357)=37.441962
W_PSNR(0.4_0.3_0.3)=37.439868
W_PSNR(1/3_1/3_1/3)=37.429806
W_MSNR(0.4_0.243_0.357)=31.129244
W_MSNR(0.4_0.3_0.3)=31.062805
W_MSNR(1/3_1/3_1/3)=31.171975
Actual_PSNR_weight(1/3_1/3_1/3)=37.429806
==========================================================================
```

```
==============================================================================
cover_image          Size_secret_data              PSNR                  MSE
                                                    dB
==============================================================================
Blue_color.jpg              1280000            37.538776             11.460478
PSNR_R=37.556763              PSNR_G=37.415684                PSNR_B=37.647019
MSNR_R=6.523007              MSNR_G=25.151299                MSNR_B=28.776125
MSNR=24.008758
W_PSNR(0.4_0.243_0.357)=37.554702
W_PSNR(0.4_0.3_0.3)=37.541516
W_PSNR(1/3_1/3_1/3)=37.539822
W_MSNR(0.4_0.243_0.357)=18.994045
W_MSNR(0.4_0.3_0.3)=18.787430
W_MSNR(1/3_1/3_1/3)=20.150144
Actual_PSNR_weight(1/3_1/3_1/3)=28.500469
==============================================================================
```

```
==============================================================================
cover_image          Size_secret_data              PSNR                  MSE
                                                    dB
==============================================================================
green_color.jpg             1280000            34.503834             23.051382
PSNR_R=33.537560              PSNR_G=37.406737                PSNR_B=33.575594
MSNR_R=-12.309591             MSNR_G=26.895562                MSNR_B=-11.808793
MSNR=14.750319
W_PSNR(0.4_0.243_0.357)=34.491348
W_PSNR(0.4_0.3_0.3)=34.709723
W_PSNR(1/3_1/3_1/3)=34.839964
W_MSNR(0.4_0.243_0.357)=-2.603954
W_MSNR(0.4_0.3_0.3)=-0.397806
W_MSNR(1/3_1/3_1/3)=0.925726
Actual_PSNR_weight(1/3_1/3_1/3)=22.395777
==============================================================================
```

```
==============================================================================
cover_image          Size_secret_data              PSNR                  MSE
                                                    dB
==============================================================================
red_color.jpg               1280000            36.762699             13.702837
PSNR_R=36.336424              PSNR_G=36.819426                PSNR_B=37.173024
MSNR_R=35.197359              MSNR_G=31.833742                MSNR_B=31.110512
MSNR=32.967158
W_PSNR(0.4_0.243_0.357)=36.752460
W_PSNR(0.4_0.3_0.3)=36.732305
W_PSNR(1/3_1/3_1/3)=36.776291
W_MSNR(0.4_0.243_0.357)=32.920996
W_MSNR(0.4_0.3_0.3)=32.962220
W_MSNR(1/3_1/3_1/3)=32.713871
Actual_PSNR_weight(1/3_1/3_1/3)=36.776291
==============================================================================
```

Embedding combination 2_1_1

```
==============================================================================
cover_image          Size_secret_data              PSNR                  MSE
                                                    dB
==============================================================================
peppers_color.jpg             1280000            37.419182             11.780459
PSNR_R=37.410364              PSNR_G=37.483883                PSNR_B=37.364136
MSNR_R=32.404398              MSNR_G=30.340731                MSNR_B=25.304570
MSNR=29.820578
W_PSNR(0.4_0.243_0.357)=37.411726
W_PSNR(0.4_0.3_0.3)=37.418552
W_PSNR(1/3_1/3_1/3)=37.419461
W_MSNR(0.4_0.243_0.357)=29.368288
W_MSNR(0.4_0.3_0.3)=29.655349
W_MSNR(1/3_1/3_1/3)=29.349900
Actual_PSNR_weight(1/3_1/3_1/3)=36.727126
==============================================================================
```

```
================================================================================
cover_image          Size_secret_data            PSNR              MSE
                                                  dB
================================================================================
lena_color.jpg               1280000           37.550079            11.430689
PSNR_R=37.353306              PSNR_G=37.651402                PSNR_B=37.652425
MSNR_R=34.334176              MSNR_G=29.438803                MSNR_B=29.991048
MSNR=31.580308
W_PSNR(0.4_0.243_0.357)=37.532529
W_PSNR(0.4_0.3_0.3)=37.532471
W_PSNR(1/3_1/3_1/3)=37.552378
W_MSNR(0.4_0.243_0.357)=31.594103
W_MSNR(0.4_0.3_0.3)=31.562625
W_MSNR(1/3_1/3_1/3)=31.254675
Actual_PSNR_weight(1/3_1/3_1/3)=36.915868
================================================================================
```

```
================================================================================
cover_image          Size_secret_data            PSNR              MSE
                                                  dB
================================================================================
barbra_color.jpg               1280000          37.562702            11.397515
PSNR_R=37.400195               PSNR_G=37.640362               PSNR_B=37.652233
MSNR_R=31.850826               MSNR_G=29.692260               MSNR_B=28.936825
MSNR=30.265671
W_PSNR(0.4_0.243_0.357)=37.548533
W_PSNR(0.4_0.3_0.3)=37.547857
W_PSNR(1/3_1/3_1/3)=37.564264
W_MSNR(0.4_0.243_0.357)=30.285996
W_MSNR(0.4_0.3_0.3)=30.329056
W_MSNR(1/3_1/3_1/3)=30.159971
Actual_PSNR_weight(1/3_1/3_1/3)=37.004286
```

```
================================================================================
cover_image          Size_secret_data            PSNR              MSE
                                                  dB
================================================================================
baboon (512 x 512)_color.jpg               1280000          37.565751          11.389517
PSNR_R=37.405435               PSNR_G=37.663087               PSNR_B=37.633330
MSNR_R=31.531799               MSNR_G=31.188577               MSNR_B=29.968074
MSNR=30.926152
W_PSNR(0.4_0.243_0.357)=37.549403
W_PSNR(0.4_0.3_0.3)=37.551099
W_PSNR(1/3_1/3_1/3)=37.567284
W_MSNR(0.4_0.243_0.357)=30.890146
W_MSNR(0.4_0.3_0.3)=30.959715
W_MSNR(1/3_1/3_1/3)=30.896150
Actual_PSNR_weight(1/3_1/3_1/3)=36.924555
```

```
================================================================================
cover_image          Size_secret_data            PSNR              MSE
                                                  dB
================================================================================
Balloon_color.jpg              1280000          37.423186            11.769602
PSNR_R=37.268900               PSNR_G=37.609580               PSNR_B=37.397868
MSNR_R=29.818751               MSNR_G=31.383479               MSNR_B=32.300626
MSNR=31.218321
W_PSNR(0.4_0.243_0.357)=37.397727
W_PSNR(0.4_0.3_0.3)=37.409795
W_PSNR(1/3_1/3_1/3)=37.425449
W_MSNR(0.4_0.243_0.357)=31.085010
W_MSNR(0.4_0.3_0.3)=31.032732
W_MSNR(1/3_1/3_1/3)=31.167619
Actual_PSNR_weight(1/3_1/3_1/3)=37.425449
================================================================================
```

```
================================================================================
cover_image          Size_secret_data            PSNR              MSE
                                                  dB
================================================================================
Blue_color.jpg               1280000           37.546540            11.440010
PSNR_R=37.328058               PSNR_G=37.673112               PSNR_B=37.647019
MSNR_R=6.294302                MSNR_G=25.408728               MSNR_B=28.776125
MSNR=24.016522
W_PSNR(0.4_0.243_0.357)=37.525775
W_PSNR(0.4_0.3_0.3)=37.527262
W_PSNR(1/3_1/3_1/3)=37.549396
W_MSNR(0.4_0.243_0.357)=18.965118
W_MSNR(0.4_0.3_0.3)=18.773176
W_MSNR(1/3_1/3_1/3)=20.159718
Actual_PSNR_weight(1/3_1/3_1/3)=28.510044
```

```
==========================================================================
cover_image          Size_secret_data            PSNR                 MSE
                                                  dB
==========================================================================
green_color.jpg              1280000            34.498175            23.081436
PSNR_R=33.425356              PSNR_G=37.657753                    PSNR_B=33.575594
MSNR_R=-12.421795              MSNR_G=27.146577                    MSNR_B=-11.808793
MSNR=14.744660
W_PSNR(0.4_0.243_0.357)=34.507464
W_PSNR(0.4_0.3_0.3)=34.740147
W_PSNR(1/3_1/3_1/3)=34.886235
W_MSNR(0.4_0.243_0.357)=-2.587839
W_MSNR(0.4_0.3_0.3)=-0.367383
W_MSNR(1/3_1/3_1/3)=0.971997
Actual_PSNR_weight(1/3_1/3_1/3)=22.442047
```

```
==========================================================================
cover_image          Size_secret_data            PSNR                 MSE
                                                  dB
==========================================================================
red_color.jpg              1280000            36.766376            13.691241
PSNR_R=36.152324              PSNR_G=37.046617                    PSNR_B=37.173024
MSNR_R=35.013259              MSNR_G=32.060933                    MSNR_B=31.110512
MSNR=32.970835
W_PSNR(0.4_0.243_0.357)=36.734027
W_PSNR(0.4_0.3_0.3)=36.726822
W_PSNR(1/3_1/3_1/3)=36.790655
W_MSNR(0.4_0.243_0.357)=32.902563
W_MSNR(0.4_0.3_0.3)=32.956737
W_MSNR(1/3_1/3_1/3)=32.728235
Actual_PSNR_weight(1/3_1/3_1/3)=36.790655
```