An Najah National University

Faculty of Graduated Studies

A Comparable Study of Hiding Information in Images Using Least Significant Bit (LSB) Substitution and Pixel Value Differencing (PVD) Methods

By

Rana Tayseer Sabbah

Supervisor

Dr. Mohammad Assad

Co- Supervisor

Dr. Loa'i Malhis

This Thesis is Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Computational Mathematics, Faculty of Graduate Studies, An-Najah National University, Nablus, Palestine.

2016

A Comparable study of Hiding Information in Images Using Least Significant Bit (LSB) Substitution and Pixel Value Differencing (PVD) Methods

By

Rana Tayseer Sabbah

This thesis was defended successfully on 11/5/2016 and approved by:

Defense Committee members	**Signature**
– Dr. Mohammad Assad / Supervisor	[illegible]
– Dr. Loa'i Malhis / Co- Supervisor	[illegible]
– Dr. Allam Mousa / External Examiner	[illegible]
– Dr. Sameer Matar / Internal Examiner	[illegible]

Dedication

This thesis is dedicated to the big hearts, Mum and Dad, for their endless love and encouragement, thank you both for giving me all the support to reach my dreams.

Acknowledgement

First and above all, I would like to thank and praise my God for all his blessings that give me the capability to proceed successfully.

Next, a heartfelt thanks to my supervisors Dr.Mohammad Assad and Dr. Lua’i Malhees for their support during my research time. I really appreciate their advice and assistance.

Thanks deeply for my external examiner Dr. Allam Mousa and my internal examiner Dr. Sameer Matar for their kind, excellent and valuable comments. To my close friends and colleagues, thank you for your encouragement. Your friendship supports me in every difficult moment.

Finally, the deepest thanks to my parents, my brothers and sisters for their love and support throughout every moment in my life, thank you for giving me the strength to continue and chase my dreams..

الإقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان

A Comparable Study of Hiding Information in Images Using Least Significant Bit (LSB) Substitution and Pixel Value Differencing (PVD) Methods

أقر بأن ما شملت عليه الرسالة هو نتاج جهدي الخاص, باستثناء ما تمت الإشارة إليه حيثما ورد, وأن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل لنيل أي درجة أو لقب علمي أو بحثي لدى أي مؤسسة علمية أو بحثية

<u>Declaration</u>

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degrees or qualifications.

Student's Name:	**اسم الطالب:**
Signature	**التوقيع:**
Date	**التاريخ:**

Table of Contents

List of Figures

List of Tables

A Comparable Study of Hiding Information in Images Using Least Significant Bit (LSB) Substitution and Pixel Value Differencing (PVD) Methods

By
Rana Tayseer Sabbah
Supervisor
Dr. Mohammad Assad
Co- Supervisor
Dr. Loa’i Malhis

Abstract

Steganography is one of the most powerful techniques that conceal the existence of hidden secret data inside a cover object.

In this thesis, we take an image for a carrier of secret data which is known as a host or cover image. After embedding the secret data into the host image, the output image of this hiding process is called a stego-image, data hiding schemes are used, by applying some pixel adjustment process to the stego-image obtained by the simple LSB substitution method, and using PVD method, the image quality of the stego-image can be greatly improved with low extra computational complexity. The mean-square-error between the stego-image and the cover-image and Peak Signal to noise ratio are computed. Experimental results show that the stego-image is visually indistinguishable from the original cover-image.

As we make a comparable study, the results show that PVD method gives a better image quality combined with the capacity of hidden data, as well as the complexity and robustness of embedded data are increased where the secret data is stored in a difficult way to understand by any intruder.

Preface

History of Data Hiding :

The idea of communicating secretly is as old as communication itself. In this section, we briefly discuss the historical development of information hiding techniques such as steganography.

Early steganography was messy. Before phones, before mail, before horses, messages were sent on foot. If you wanted to hide a message, you had two choices: have the messenger memorize it, or hide it on the messenger. While information hiding techniques have received a tremendous attention recently, its application goes back to Greek times. According to Greek historian Herodotus, the famous Greek tyrant Histiaeus, while in prison, used unusual method to send message to his son-in-law. He shaved the head of a slave to tattoo a message on his scalp. Histiaeus then waited until the hair grew back on slave’s head prior to sending him off to his son-in-law. The second story also came from Herodotus, which claims that a soldier named Demeratus needed to send a message to Sparta that Xerxes intended to invade Greece. Back then, the writing medium was written on wax-covered tablet. Demeratus removed the wax from the tablet, wrote the secret message on the underlying wood, recovered the tablet with wax to make it appear as a blank tablet and finally sent the document without being detected. Invisible inks have always been a popular method of steganography. Ancient Romans used to write between lines using invisible inks based on available substances such as fruit juice and milk. When heated, the invisible inks would darken, and become legible. Ovid in his “Art of Love” suggests using milk to write invisibly. Later chemically affected sympathetic inks were developed. Invisible inks were used as recently as World War II. Modern invisible inks fluoresce under ultraviolet light and are used as anti-counterfeit devices. For example, "VOID" is printed on checks and other official documents in an ink that appears under the strong ultraviolet light used for photocopies [1].

Chapter One

Introduction to Data Hiding

Chapter One

1.1 Introduction:

In our thesis, data hiding schemes are used, by applying some pixel adjustment process to the stego-image obtained by the simple LSB substitution method, and using PVD method, the image quality of the stego-image can be greatly improved with low extra computational complexity. The mean-square-error between the stego-image and the cover-image and Peak Signal to noise ratio are computed. Experimental results show that the stego-image is visually indistinguishable from the original cover-image.

In the first chapter we give an introduction to data hiding and steganography, next we explain the way of using LSB in image hiding, then we briefly talk about PVD method and compare it with LSB method. Then Dynamic LSB is used with examples to show its goals. Finally, we give the conclusion and results that compare the several methods of hiding information in images which we talk about. ***Matlab program is used to convert the Algorithms of data hiding techniques that we discuss into codes which return good results that help in the comparison we make.***

1.2 Data Hiding:

In the Internet, various kinds of data are sent, transmitted and received every single moment. Some of them may be secret information of commerce and others confidential messages from the government, both are candidate preys for grabbers to access. In order to keep the grabbers away, a variety of techniques has been proposed. One of the most famous methods is **data**

encryption, which uses a certain algorithm to transform data into cipher texts. Only the user that has keys can decrypt the secret data from the cipher texts. For any grabber who does not have a key, the cipher texts will look like nothing but streams of meaningless codes. Although data encryption is a good way to prevent grabbers from accessing secret data, it still has some weaknesses. The appearance of cipher texts would give grabbers an impulse to recover them. Moreover, grabbers might even simply destroy the cipher texts when they have trouble in recovering them so that the legal receivers cannot get the data in time. That is the reason why **data hiding** has been researched recently [1].

Data hiding techniques embed the important data into multimedia data such as images, videos or movies.

In our thesis, we take an image as a carrier of secret data which is known as a host image or cover image. After embedding the secret data into the cover image, the output image of this hiding process is called a stego image.

When we discuss image hiding, one of the most important things is that the quality of the stego image must not be degraded too much after embedding. The goal of data hiding is to make the secret message invisible to grabbers. Thus, if the process of embedding degrades the quality of the stego image too much, any grabber will easily take notice of it.

A Data Hiding method replaces some least significant bits of the host image with the secret data. After the hiding process, the researches of data hiding focus on the difficulty of removing the embedded data through operations of images. Most researchers concerns are concentrated on watermarking.

In our opinion, one of the important things in data hiding is the embedding capacity. When we want to transmit important secret data via a certain technique of data hiding, it is obviously not practical if we have no choice but to break the secret data into pieces and put each piece in a large host image due to the limited capacity.

On the other hand, some Data Hiding techniques directly replaces some bits of each pixel value in the host image with the secret data; it does not focus on the difficulty of removing secret data through images operations but rather on the capacity of embedding. When talking about the capacity of embedding, the major concern is to make stego image extremely hard for grabbers to sense the existence of the secret data.

1.3 Goals of Steganography:

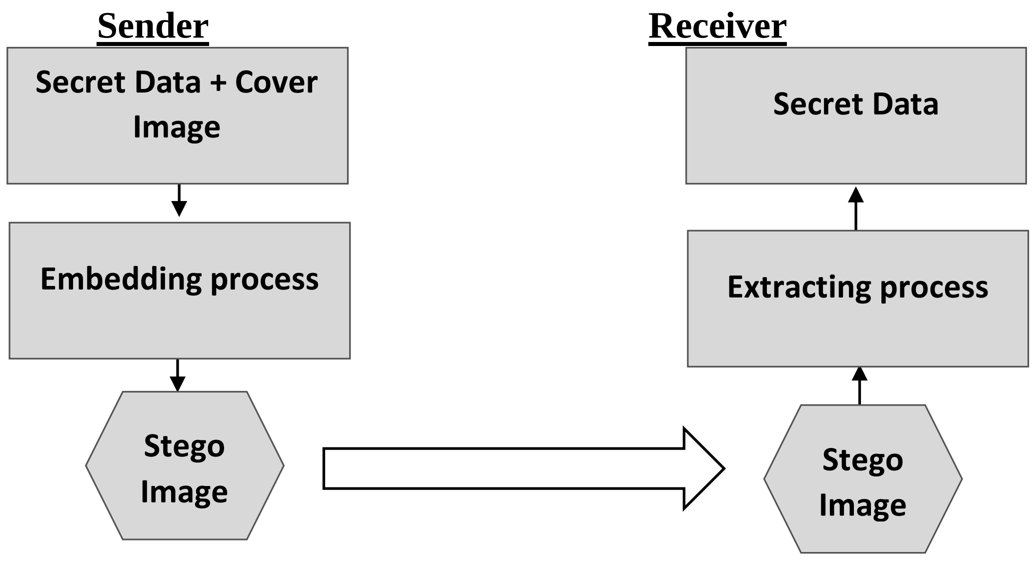


Figure (1. 1): General Steganography Mechanism

Steganography or Stego as it often referred to in the IT community, literally means, “Covered writing” which is derived from the Greek language. Steganography is defined as follows, ***“The art and science of Communicating in a way which hides the existence of the communication”***.

The goal of Steganography is to hide messages inside other messages in a way that does not allow any enemy to even detect that there is a second message present. In a digital world, Steganography and Cryptography [2] [3] are both intended to protect information from unwanted parties. Both Steganography and Cryptography are excellent means by which to accomplish this but neither technology alone is perfect and both can be broken. For this reason, most experts suggest using both to add multiple layers of security [20] [21].

A few key properties that must be considered when creating a data hiding system are:

- Imperceptible: the property in which a person should be unable to distinguish the original and the stego image.
- Embedding Capacity: Refers to the amount of secret information that can be embedded without degradation of the quality of the image.
- Robustness: Refers to the degree of difficulty required to destroy embedded information without destroying the cover image.

1.4 Steganography vs. Cryptography:

Basically, the purpose of cryptography and steganography is to provide secret communication. However, steganography is not the same as cryptography.

Cryptography hides the **contents** of a secret message from a malicious people, whereas steganography even **conceals the existence** of the message. Steganography must not be confused with cryptography where we transform the message so as to make its meaning obscure to a malicious people who intercept it.

Therefore, the definition of breaking the system is different [3]. In cryptography, the system is broken when the attacker can read the secret message. Breaking a steganographic system needs the attacker to detect that steganography has been used and he is able to read the embedded message.

In cryptography, the structure of a message is scrambled to make it meaningless and unintelligible unless the decryption key is available. It makes no attempt to hide the encoded message. Basically, cryptography offers the ability of transmitting information between persons in a way that prevents a third party from reading it.

In contrast, Steganography does not alter the structure of the secret message, but hides it inside a *cover image* so it cannot be seen. A message in cipher text, for instance, might arise suspicion on the part of the recipient while an "invisible" message created with stenographic methods will not. In other word, steganography prevents an unintended recipient from suspecting that the data exists.

It is possible to combine the techniques by encrypting message using cryptography and then hiding the encrypted message using steganography. The resulting *stego image* can be transmitted without revealing that secret information is being exchanged. Furthermore, even if the attackers were to defeat the steganographic technique and detect the message from the *stego* *object*, they would still require the cryptographic decoding key to decipher the encrypted message.

There are many applications for digital steganography of image, including copyright protection, feature tagging, and secret communication.

Copyright notice or watermark can be embedded inside an image to identify it as intellectual property. If someone attempts to use this image without permission, we can prove by extracting the watermark [4].

1.5 Steganographic Techniques:

Over the past few years, numerous steganography techniques that embed hidden messages in multimedia objects have been proposed [5]. There have been many techniques for hiding information or messages in images in such a manner that the alterations made to the image are unrealizable. Common approaches are include:

- Least significant bit insertion (LSB)
- Masking and filtering
- Transform techniques

Least significant bits (LSB) insertion is a simple approach for embedding information in image file. The simplest steganographic techniques embed the bits of the message directly into least significant bit plane of the *cover image* in a deterministic sequence. Modulating the least significant bit does not result in human perceptible difference because the amplitude of the change is small.

Masking and filtering techniques, usually restricted to 24 bits and gray scale images, hide information by marking an image, in a manner similar to paper watermarks. The techniques perform analysis of the image, thus embed the information in significant areas so that the hidden message is more integral to the cover image than just hiding it in the noise level.

Transform techniques embed the message by modulating coefficients in a transform domain, such as the *Discrete Cosine Transform* (DCT) used in JPEG compression, *Discrete Fourier Transform*, or *Wavelet Transform*. These methods hide messages in significant areas of the *cover-image*, which make them more robust to attack. Transformations can be applied over the entire image, to block throughout the image, or other variants [4].

1.6 Image Pixels:

All computer-based images are composed of an array of dots, called pixels that make a very fine grid. Each one of these pixels has its own color, represented internally as separate quantities of red, green and blue. And each color’s level ranges between 0 (none of the color) and 255 (a full amount of the color).

1.7 Process of Producing a Stego-Image:

The *cover image* will be combined with the message. This will produce the output called *stego image*. The *stego image* will appear identical to the *cover image* by embedding the message into the *cover image* without supplying any password or *stego-key*.

We have to understand the way of LSB and PVD steganographic techniques to insert the message into the *cover image* and extract the message from the *stego image* produced [19].

Chapter Two

Least Significant Bit Substitution Method (LSB)

Chapter Two

2.1 Introduction to LSB Substitution:

Nowadays Internet has become the most popular communication media for message transmission in different places of the world. Two schemes are used to protect secret messages from being captured during transmission. One of them is encryption where the secret information is encoded in another form by using a secret key before sending, which can only be decoded with secret keys. The most popular encryption techniques are DES, RSA etc. Other way is Steganography which is a technique of hiding secret information into a cover media or carrier. If the cover media is a digital image, it is called *cover image* and the cover image with hidden data is called *stego image*. Steganographic techniques can be used in military, commercial, anti-criminal and so on.

There are various steganographic techniques available where a digital image is used as a carrier. The most common and simplest method is least-significant-bit (LSB) substitution, where the LSB of each pixel of the cover image is replaced by one bit of the secret data [5].

The LSB is the lowest significant bit in the byte value of the pixel. The LSB based image steganography embeds the secret message in the least significant bits of pixel values of the cover image.

The concept of LSB Embedding is simple. It exploits the fact that the level of precision in many image formats is far greater than that perceivable by average human vision. Therefore, an altered image with slight variations will be indistinguishable from the original image by a human being, just by looking at it.

In conventional, LSB technique requires eight bytes of pixels to store 1 byte of secret data [6].

The least significant bits of the cover image are used to conceal the message. The simplest of the LSB steganographic techniques is to use only one LSB's replacement at each pixel's value. LSB replacement steganography flips the last bit of each of the data values to reflect the message that needs to be hidden.

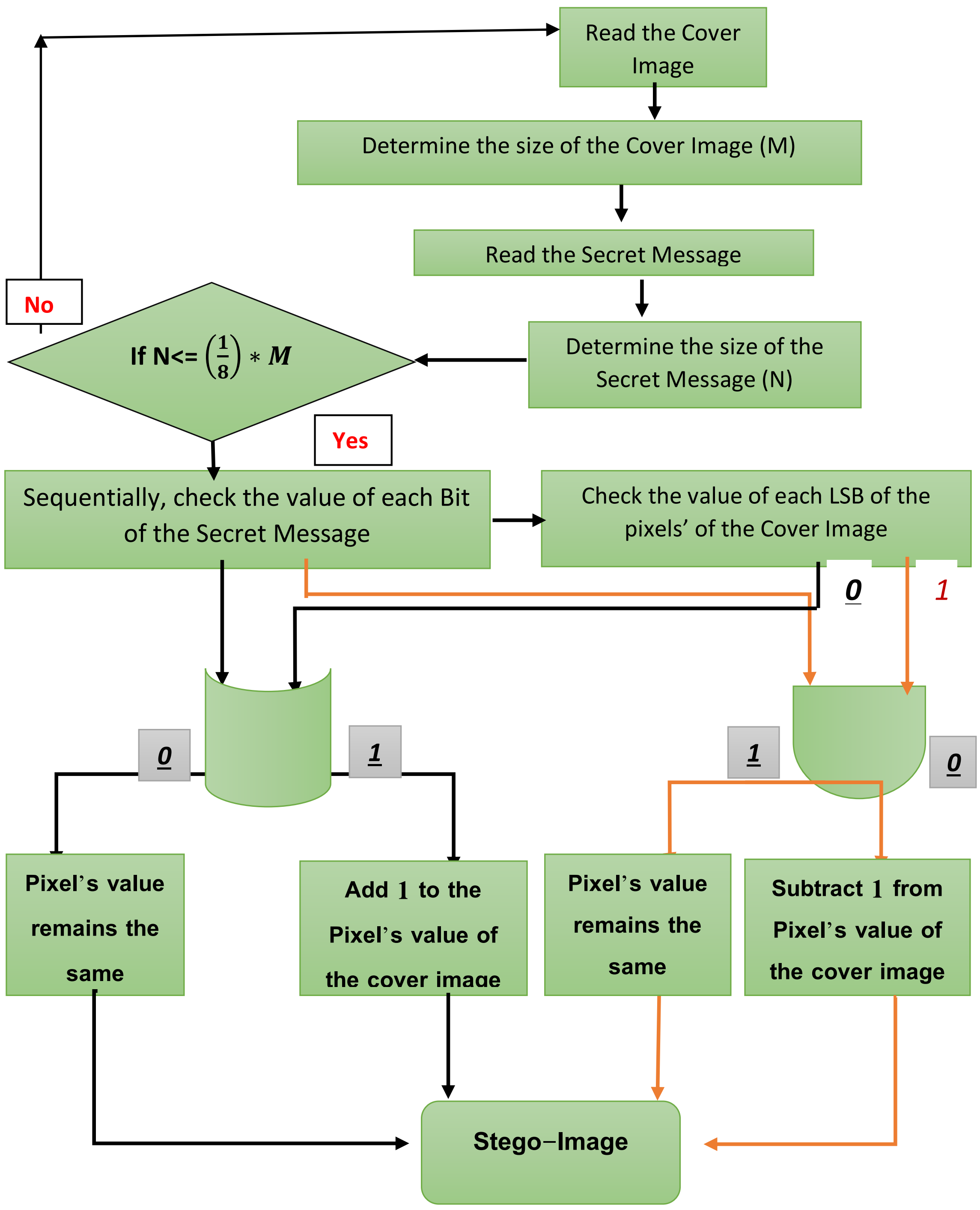
2.2 LSB Algorithm:

Inputs: Cover Image, Secret Message

Procedure:

```
Step1: Read the Cover Image.
Step2: Decide the size of the cover image = M.
Step3: Read the Secret message.
Step4: Decide the size of the secret message= N.
Step5: check whether (N <=$\left(\frac{1}{8}\right) * M$)
Step6: A) If the pixel of the cover image is odd → check the Secret Bit to be hidden:
    • If 0 → subtract 1 from Pixel's value of the cover image.
    • Else if 1 → pixel's value of the cover image remains the same.
    B) Else if the pixel's value of the cover image is even → check the Secret Bit to be hidden:
    • If 0 → Pixel's value of the cover image remains the same.
    • Else if 1 → Add 1 to the Pixel's value of the cover image.
Step7: Show the image after embedding and notice the difference if appears.
End.
```

Output: Stego-image.



Read the Cover Image
Determine the size of the Cover Image (M)
Read the Secret Message
Determine the size of the Secret Message (N)
If N<= $\left(\frac{1}{8}\right) * M$
No
Yes
Sequentially, check the value of each Bit of the Secret Message
Check the value of each LSB of the pixels' of the Cover Image
0
1
0
1
1
0
Pixel's value remains the same
Add 1 to the Pixel's value of the cover image
Pixel's value remains the same
Subtract 1 from Pixel's value of the cover image
Stego−Image

Figure (2. 1): The Flow Chart of LSB Substitution Method

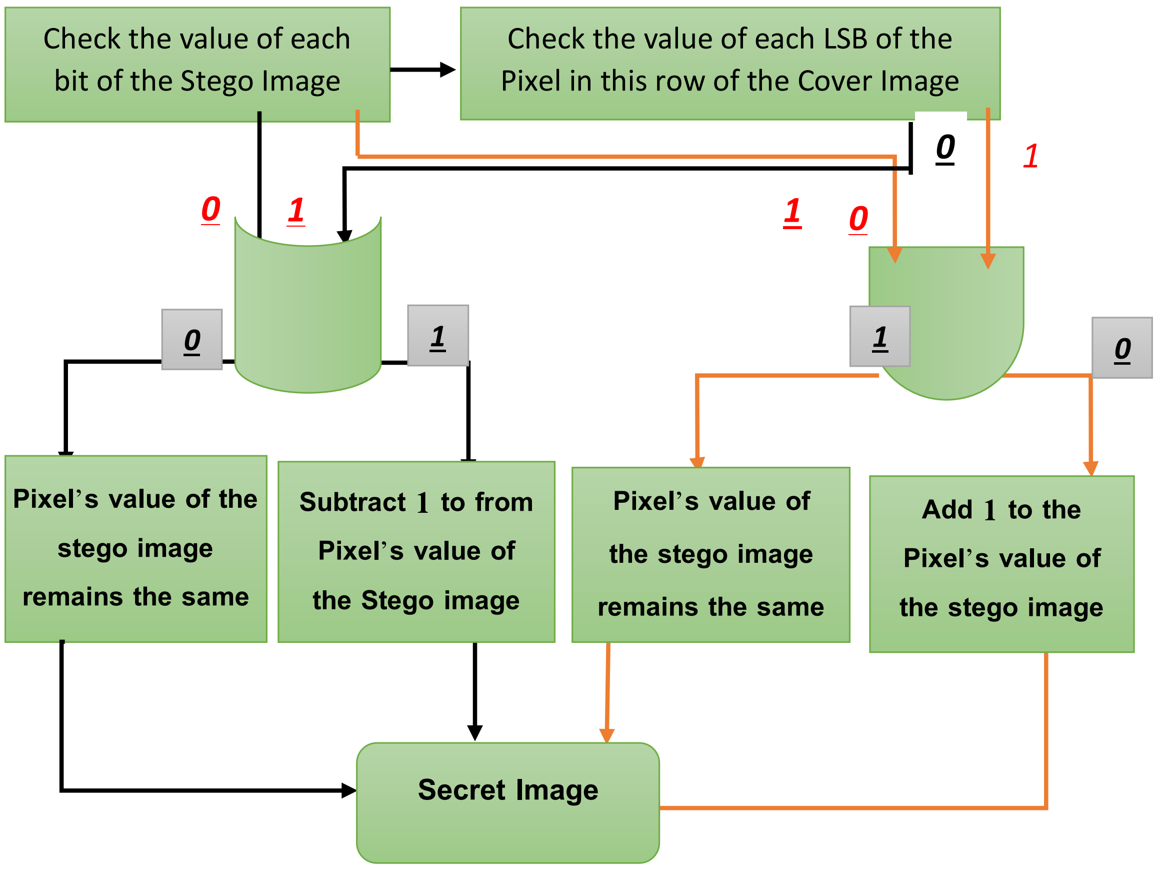

Check the value of each bit of the Stego Image
Check the value of each LSB of the Pixel in this row of the Cover Image
0
1
0
1
1
0
0
1
1
0
Pixel's value of the stego image remains the same
Subtract 1 to from Pixel's value of the Stego image
Pixel's value of the stego image remains the same
Add 1 to the Pixel's value of the stego image
Secret Image

Figure (2. 2): The Recover Flow Chart of LSB Substitution Method

2.3 Examples of using LSB in hiding information:

To illustrate LSB technique, we provide the following examples: Consider an 8-bit grayscale bitmap image where each pixel is stored as a byte representing a grayscale value.

Suppose that the first eight pixels of the original image have the following grayscale values:

1101001**0** 0100101**0** 1001011**1** 1000110**0**

0001010**1** 0101011**1** 0010011**0** 0100001**1**

To hide the letter "C" whose binary value is 01000011 (decimal value is 67), we would replace the LSBs of these pixels to have the following new grayscale values:

1101001**0** 0100101**1** 1001011**0** 1000110**0**

0001010**0** 0101011**0** 0010011**1** 0100001**1**

Note that, on average, about half of the LSBs need to be changed. The difference between the cover (i.e. original) image and the stego image will be hardly noticeable to the human eye.

LSB steganography, as described above, replaces the LSBs of data values to match bits of the message. It can equally alter the data value by a small amount, ensuring that the legal range of data values is preserved.

- From the above example we can infer that 1-LSB insertion usually has a 50% chance to change a LSB every 8 bits, thus adding very little alteration to the original image.

2.4 The advantages and disadvantages of LSB:

Advantages:

- There is a less chance for degradation of the original image: Modulating the LSB does not result in a human-perceptible difference because the amplitude of the change is small. Therefore, to the human eye, the resulting *stego-image* will look identical to the *cover-image*. This allows high perceptual transparency of LSB.
- Simple: simplicity to embed the bits of the message directly into the LSB plane of the *cover-image.*

Disadvantages:

- Less robust, the hidden data can be lost with image manipulation:

  It is very sensitive to any kind of filtering or manipulation of the *stego-image*. Scaling, rotation, cropping, addition of noise, or lossy compression to the *stego-image* will destroy the message.
- Limited Capacity: for the hiding capacity, the size of information to be hidden relatively depends on the size of the *cover-image*. The simple LSB method limits the size of the secret data to (1/8) of the size of the cover image.
- Hidden data can be easily detected by simple attacks: an attacker can easily destruct the message by removing or vanishing the entire LSB plane with very little change in the perceptual quality of the modified *stego image*. Therefore, if this method causes someone to suspect something hidden in the *stego image*, then the method is not success.
- Requirement of high transmission rate due to large size of stego image.

2.5 n-LSBs Substation Method:

** **To increase the capacity of hidden information:**

LSB steganography (using least ***n*-bits)** to increase the capacity of the secret information to be hidden, to $n/8$ of the size of the cover image. However, increasing n distorts stego-image. In each run, we embed random data in the n least significant bits, where $1 \leq n \leq 8$. However, we need to introduce the methods to measure the quality and distortion in images.

To measure the imperceptibility of steganography several metrics are used. The metrics indicates how similar (or different) the stego-image compared with cover image [7] [8].

2.6 Performance evaluation metrics:

**** The following metrics are used:**

Mean Squared Error (MSE): is computed by performing byte by byte comparisons of the cover and stego images. The computation can be expressed as follows:

$$\boldsymbol{MSE} = \frac{1}{M*N}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(C(i,j) - S(i,j)\right)^2$$

Where:

C(i,j),S(i,j): the image pixel values before and after embedding respectively.

M*N: the size of the image.

**** Higher value of MSE indicates dissimilarity between compared images.**

MSE can be used to calculate the ***peak signal to noise ratio*** as follows:

$$\boldsymbol{PSNR} = 10 * \log_{10}\left(\frac{255^2}{MSE}\right)$$

Peak signal-to-noise ratio measures in decibels the quality of the *stego image* compared with the *cover image* [25].

The higher PSNR the better the quality of stego-image.

2.7 Examples and Results:

In our study, several *n* LSBs steganography techniques were implemented, where 1≤*n* ≤8 using the 'rice.tif' image as the cover image, and 'cameraman.tif' as the secret message (both have the same size).

The image metrics were computed for the images across the various LSB experiments.

The result *stego images* are shown in Figure 2. For 8-bit LSB, the image is nearly distorted. The results of the image metrics are summarized in the following table:

Table (2. 1):Image Metrics Results for n-LSBs Method (Both cover and hidden images have the same size)

n-bit LSB	MSE	PSNR
1-bit	0.4983	51.1561
2-bit	1.9874	45.1479
3-bit	7.9998	39.1000
4-bit	31.7432	33.1143
5-bit	122.9219	27.2345
6-bit	631.5625	20.1266
7-bit	2252.5	14.6042
8-bit	6211.3	10.1990

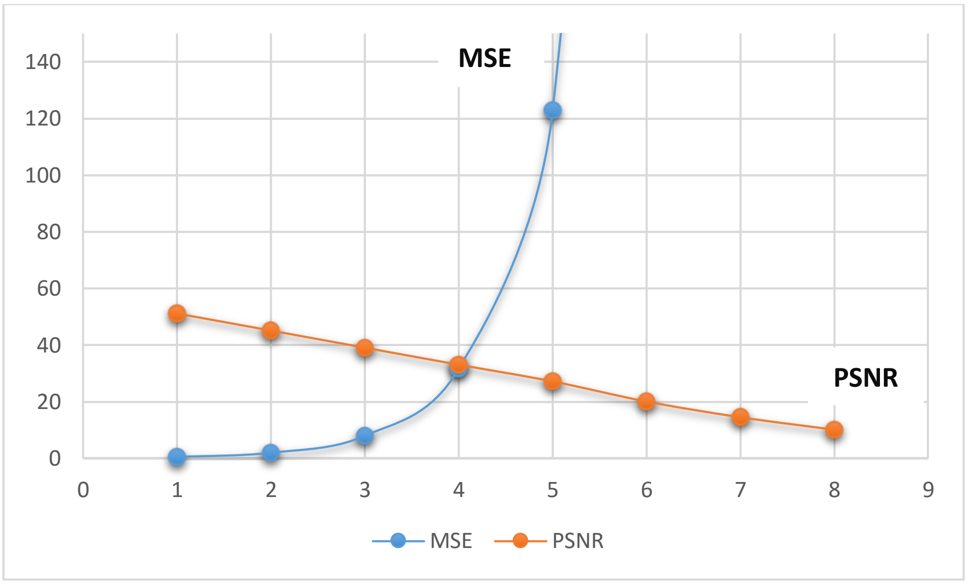


Figure (2. 3): Image Metrics Results for n-LSBs Method (Both cover and hidden images have the same size)

The figure above shows that as the number of bits to be hidden increases, the value of MSE increases, while that for PSNR decreases, so that the quality of the stego image decreases.

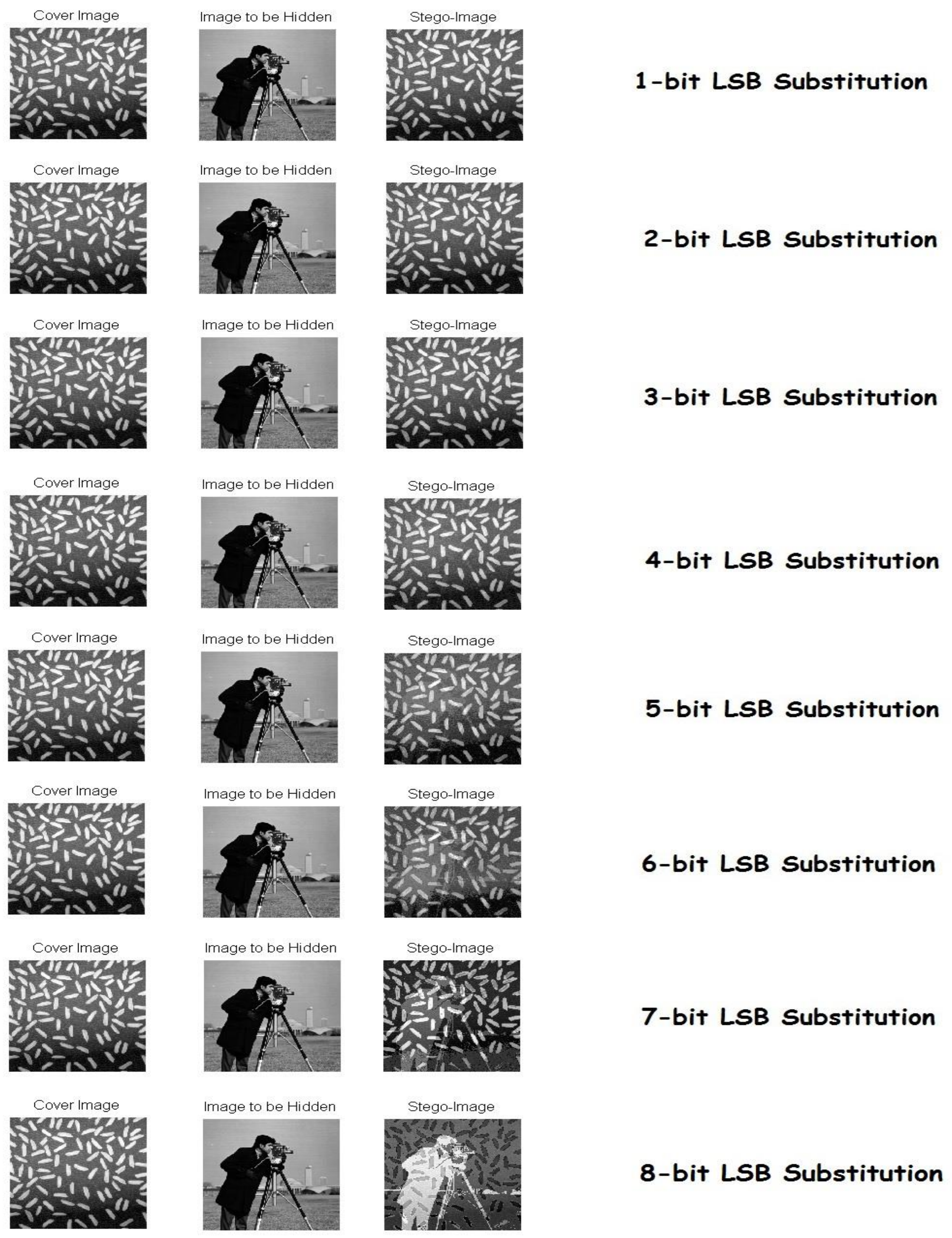


Figure (2. 4): Results of N-LSBs Substitution Method (Both cover and hidden images have the same size)

2.8 Different size of Images - Matching the size of the cover image with that of the hidden image:

Before hiding an image, note that, we have to compare the sizes of both cover and hidden images. So that:

- If the image to be hidden is bigger than the original image, scale it down.
- Tile the hidden Image, if it's smaller, so that it will cover the original image. i.e. resize the hidden image to match the size of the cover image.

Examples:

1) Tiling the Hidden Image:

Why to use tiling? *We need to enlarge the size of the hidden image to match the size of the cover image.*

We read two images and check the size of both of them:

Cover image ***(Moon.tif)***:

```
coverImage = imread('moon.tif');
% Get the number of rows and columns in the cover image.
[visibleRows visibleColumn] = size(coverImage);
```

Results are:

```
visibleRows = 537
visibleColumns = 358
```

Image to be hidden ***(Cameraman.tif)***:

```
hiddenImage = imread('cameraman.tif');
% Get the number of rows and columns in the hidden image.
[hidden Rows hidden Columns] = size (hiddenImage);
```

Results are:

```
hiddenRows = 256
hiddenColumns = 256
```

From the above results, note that the size of the hidden image is smaller than that of the cover image; so, we need to tile the hidden image to match the size of the cover image.

To tile the hidden image, we use the following Matlab code:

```
if hiddenRows < visibleRows || hiddenColumns < visibleColumns
   watermark = zeros(size(originalImage), 'uint8');
   for column = 1:visibleColumns
      for row = 1:visibleRows
         watermark(row, column) = binaryImage(mod(row,hiddenRows)+1,
mod(column,hiddenColumns)+1);
      end
   end
```

The results are shown in the following figure:

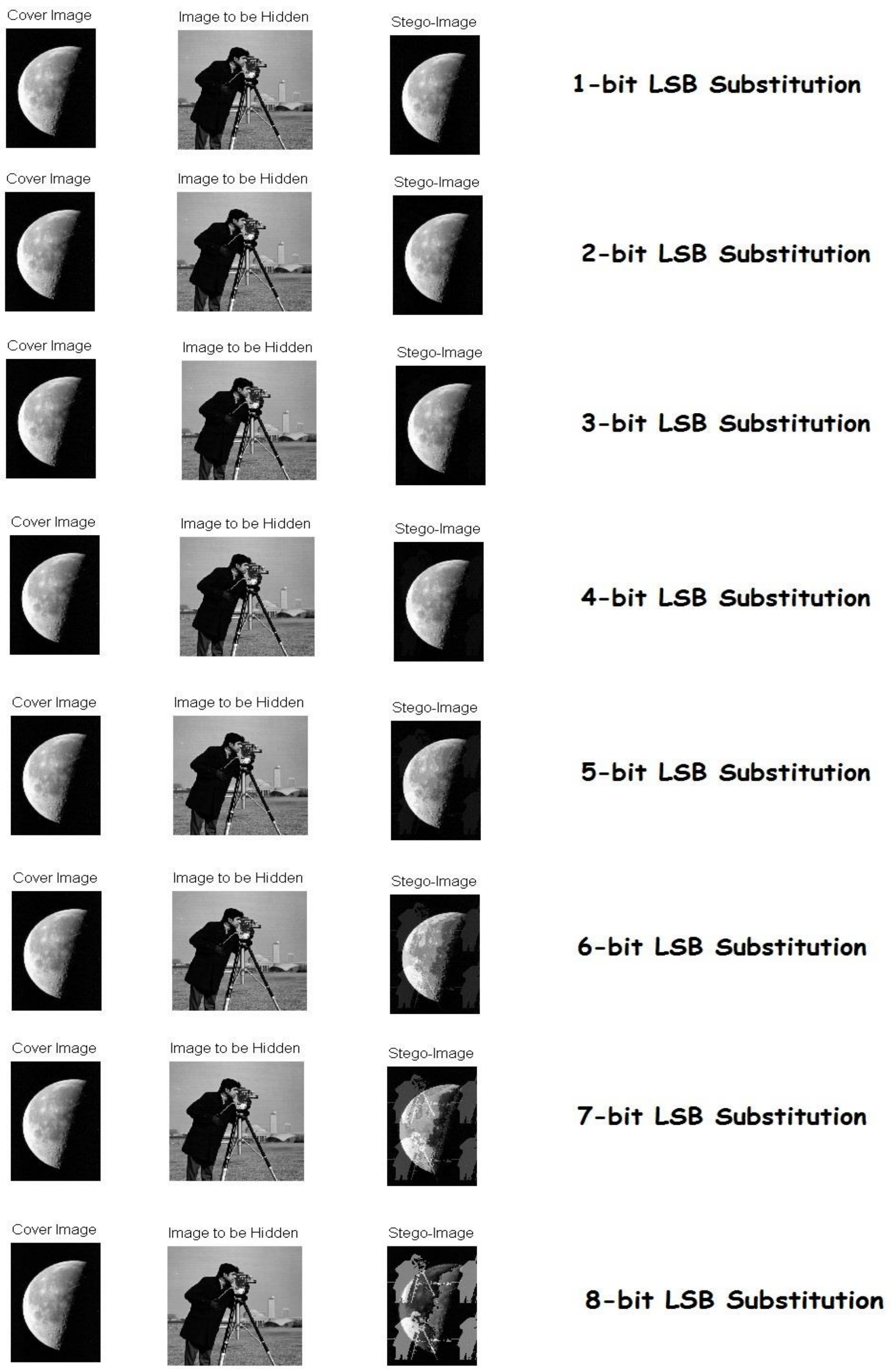


Figure (2. 5): Results of N-LSBs Substitution Method (the size of the cover image is greater than the size of the hidden image)

Table (2. 2): Image Metrics Results for n-LSBs Method (the size of the cover image is greater than the size of the hidden image)

n-bit LSB	MSE	PSNR
1-bit	0.5053	51.0952
2-bit	2.0342	45.0469
3-bit	7.1093	39.6125
4-bit	25.6724	34.0361
5-bit	97.8520	28.2251
6-bit	400.2072	22.1080
7-bit	1508	16.3468
8-bit	6364.4	10.0933

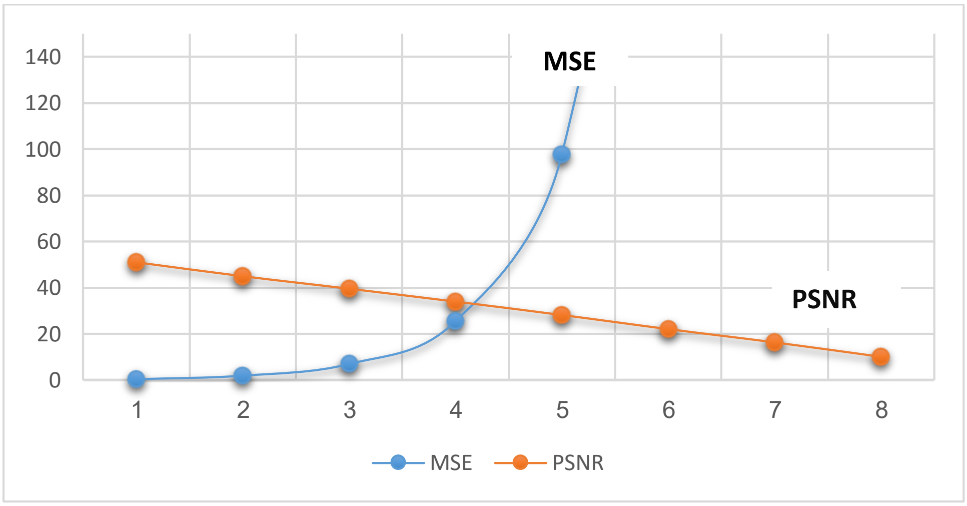


Figure (2. 6): Image Metrics results for N-LSBs Substitution method (the size of the cover image is greater than the size of the hidden image)

Fig.7 shows that as the number of bits to be hidden increases, the value of MSE increases, while that for PSNR decreases, so that the quality of the stego image decreases.

2) Scaling Down the size of the Hidden Image:

We read two images and check the size of both of them:

Cover image ***(cell.tif)***:

```
coverImage = imread('cell.tif');
% Get the number of rows and columns in the cover image.
[visibleRows visibleColumn] = size(coverImage);
```

Results are:

```
visibleRows = 159
visibleColumns = 191
```

Image to be hidden ***(coins.png)***:

```
hiddenImage = imread('coins.png');
% Get the number of rows and columns in the hidden image.
[hiddenRows hiddenColumns] = size(hiddenImage);
```

Results are:

```
hiddenRows = 246
hiddenColumns = 300
```

From the above results, note that the size of the hidden image is greater than that of the cover image; so, we need to scale down the size of the hidden image to match the size of the cover image.

To scale down the hidden image, we use a ratio that is shown in the following Matlab code:

```
if hiddenRows > visibleRows || hiddenColumns > visibleColumns
   amountToShrink = min([visibleRows / hiddenRows, visibleColumns / hiddenColumns]);
   binaryImage = imresize(binaryImage, amountToShrink);
   % Need to update the number of rows and columns.
   [hiddenRows hiddenColumns] = size(binaryImage);
End
```

The results are shown in the following figure:

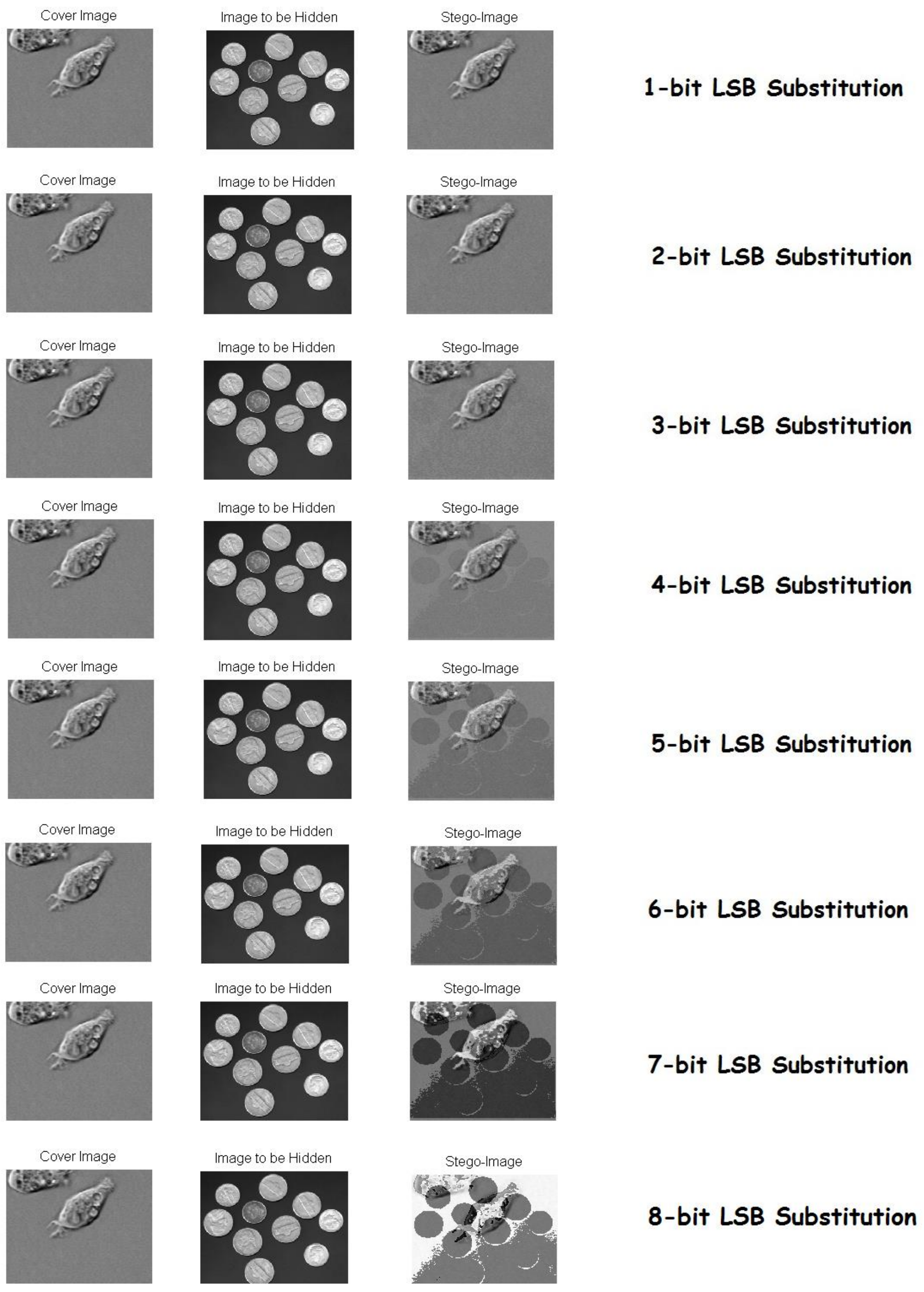


Figure (2. 7): Results of N-LSBs Substitution Method (the size of the cover image is less than the size of the hidden image)

Table (2. 3): Image Metrics Results for n-LSBs Method (the size of the cover image is less than the size of the hidden image)

n-bit LSB	MSE	PSNR
1-bit	0.5021	51.1227
2-bit	1.9931	45.1355
3-bit	7.9718	39.1152
4-bit	32.3467	33.0325
5-bit	166.7890	25.9091
6-bit	667.7630	19.8846
7-bit	2664.2	13.8752
8-bit	5672.3	10.5932

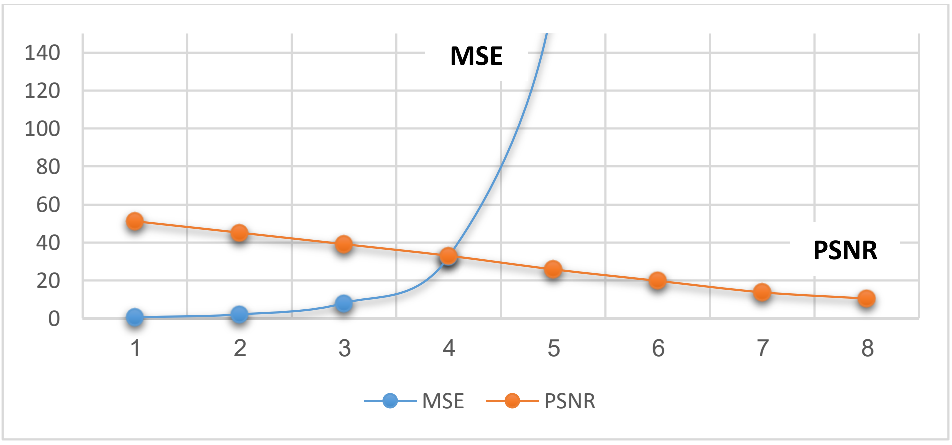


Figure (2. 8): Image Metrics results for N-LSBs Substitution method (the size of the cover image is less than the size of the hidden image)

Fig.9 shows that as the number of bits to be hidden increases, the value of MSE increases, while that for PSNR decreases, so that the quality of the stego image decreases.

Chapter Three

Pixel Value Differencing Method

Chapter Three

3.1 Introduction to Pixel Value Differencing Method (PVD):

The word "pixel" was first published in 1965 by Frederic C. Billingsley of JPL, to describe the image elements of video images from space probes to the Moon and Mars.

In digital imaging, a pixel, pel, or image element is a physical point in a raster image, or the smallest addressable element of all points in an addressable display device; so it is the smallest controllable element of an image represented on the screen. The address of a pixel corresponds to its physical coordinates.

Each pixel is a sample of an original image; more samples typically provide more accurate representations of the original. The intensity of each pixel is variable. In color image systems, a color is typically represented by three or four component intensities such as red, green, and blue, or cyan, magenta, yellow, and black.

In a gray scale image the pixel value ranges from 0 to 255. But when we use pixel-value differencing (PVD) method as image steganographic scheme, the pixel values in the stego-image may exceed gray scale range.

PVD method is used and check whether the pixel value exceeds the range on embedding. Positions where the pixel exceeds boundary have been marked and a delicate handle is used to keep the value within the range [14] [26].

3.2 The Pixel Value Differencing Method Concept:

The PVD method is proposed by Wu and Tsai can successfully provide both high embedding capacity and outstanding imperceptibility for the stego-images.

The pixel-value differencing (PVD) scheme uses the difference value between two consecutive pixels in a block to determine how many secret bits should be embedded. There are two types of the quantization range table in Wu and Tasi's method. The first was based on selecting the range widths of [8, 8, 16, 32, 64, 128], to provide large capacity. The second was based on selecting the range widths of [2, 2, 4, 4, 4, 8, 8, 16, 16, 32, 32, 64, 64], to provide high imperceptibility. Most of the related studies focus on increasing the capacity using LSB and the readjustment process, so their approach is too conformable to the LSB approach [13] [15].

In a gray scale image the pixel value ranges from 0 to 255. But when we use pixel-value differencing (PVD) method as image steganographic scheme, the pixel values in the stego-image may exceed gray scale range.

The PVD method divides the cover image into non overlapping blocks containing two connecting pixels and modifies the pixel difference in each block (pair) for data embedding.

To estimate how many secret bits will be embedded into pixel, the difference value between two, three or four pixels close to the target pixel is calculated as shown in Fig(9) [17] [18].

G(x-1,y-1) Top left pixel	G(x-1,y) Top pixel
G(x,y-1) Left pixel	G(x,y) Target pixel

Figure (3. 1): A target pixel with 3 neighboring pixels

PVD is designed in such a way that the pixel modification does not violate gray scale range interval. The selection of the range intervals is based on the characteristics of human vision sensitivity to gray value (0-255) varies from smoothness to contrast. It provides an easy way to produce a more imperceptible result than simple LSB replacement methods [9] [10] [11].

The embedded secret message can be extracted from the resulting stego-image without referencing the original cover image. Moreover, to achieve secrecy protection of hidden data a pseudo-random mechanism may be used [3]. If secret data is stored randomly it is difficult to understand by the intruder.

PVD embedding is used to increase image quality and capacity. It is also used to hide messages into gray scale as well as in color images [22] [23] [24].

3.3 Steps of Pixel Value Differencing (PVD) Method:

1) In the embedding process of a secret message, a cover image is partitioned into non-overlapping blocks of two consecutive pixels.
2) A difference value is calculated from these values of the two pixels in each block.
3) All possible difference values are classified into a number of ranges.

4) The calculated difference value then replaced by a new value to embed the value of a sub-stream of the secret message.
5) The number of bits which can be embedded in a pair of pixels is decided by the width of the range that the difference value belongs to [16].

3.4 Advantages of PVD method:

The PVD method was proposed to hide secret messages into 256 gray-valued images. It can embed large amount of data without much degradation in the image quality and thus are hardly noticeable by human eyes. It is based on the fact that human eyes can easily observe small changes in the gray values of smooth areas in the image but they cannot observe relatively larger changes at the edge areas.

3.5 The PVD Algorithm:

Inputs: Cover Image, Secret Message

Procedure:

```
Step1: Read the cover image.
Step2: Store the image in matrix M, m=number of pixels of matrix M.
Step3: Compute the difference between each two non-overlapping blocks.
Step4: K=0.
        For i=0 to m
        Diff(k)= M(i+1)-M(i)
        i= i+2
        k=k+1
```

Step5: check the value of the Difference to determine H (H = number of bits to be hidden in each pixel's block):

```
If $0 < Diff(k) \leq 16 \rightarrow H = 3$
Else if $16 < Diff(k) \leq 32 \rightarrow H = 4$
Else if $32 < Diff(k) \leq 64 \rightarrow H = 5$
Else if $64 < Diff(k) \leq 128 \rightarrow H = 6$
Else if $128 < Diff(k) \leq 255 \rightarrow H = 7$
```

Step6: Read the secret message or image and store it in matrix S, s=number of bits of the secret message (i.e. size of matrix S)

```
Step7: B=[0 0 0 0 0 0 …. 0] , size of B=H
Step8: For i=0 to H
        B(i)=S(i)
        Bd=Decimal B (convert B from binary to decimal)
```

Step9: Calculate the new difference:

If 0 < Diff(k) < 8 then new Diff = Bd (the lower value of the interval =0)

Else new Diff = Bd + 2^{H}

check the value of the new difference:

If new diff < old diff

New diff =Bd + $2^{(H+1)}$

comp = new Diff – old Diff

L1=floor ($\frac{comp}{2}$)

L1=ceil ($\frac{comp}{2}$)

M(i) = M(i) – L1

M(i+1) = M(i+1) + L2

Output: ***Stego Image***

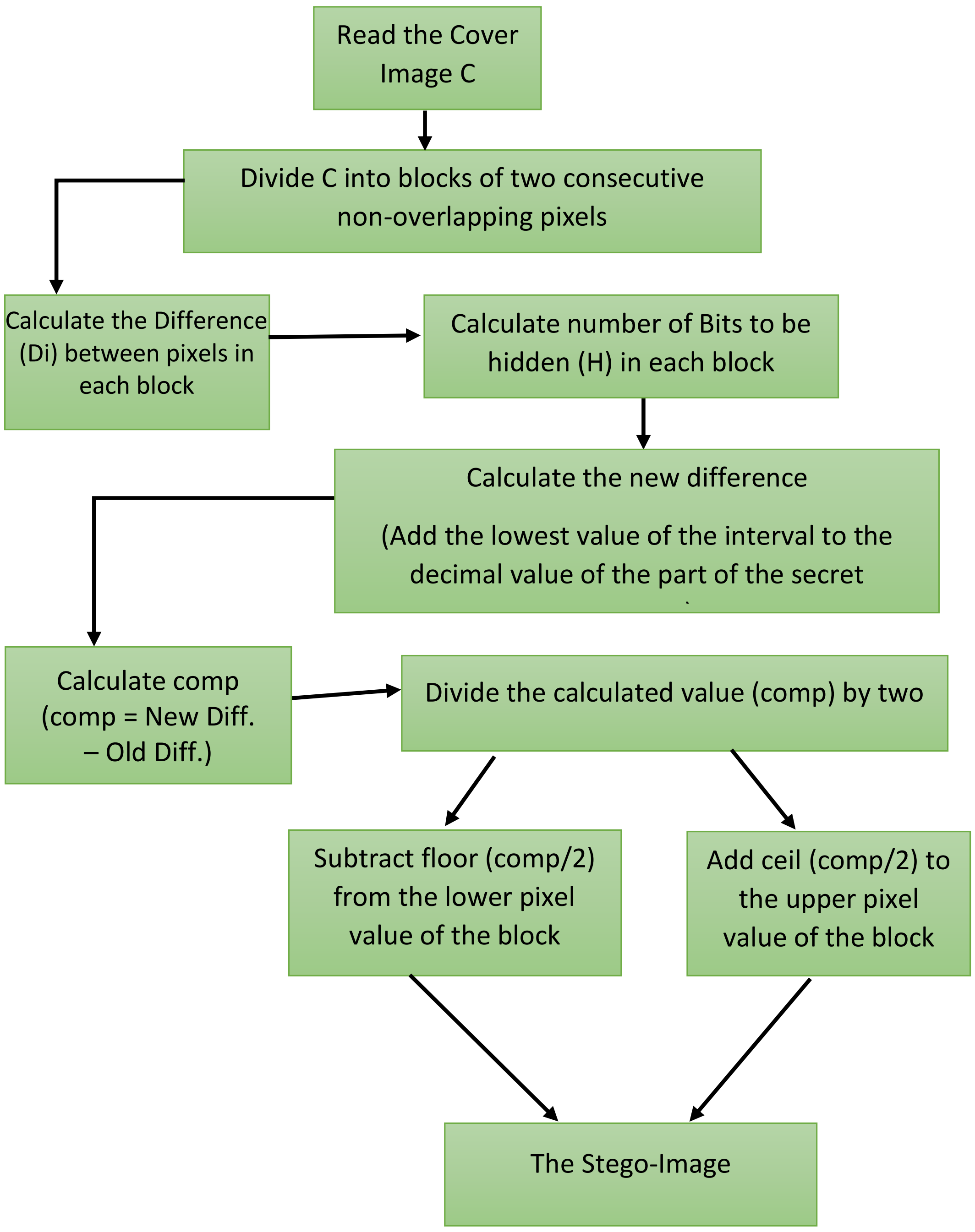

Read the Cover Image C
Divide C into blocks of two consecutive non-overlapping pixels
Calculate the Difference (Di) between pixels in each block
Calculate number of Bits to be hidden (H) in each block
Calculate the new difference
(Add the lowest value of the interval to the decimal value of the part of the secret
Calculate comp (comp = New Diff. – Old Diff.)
Divide the calculated value (comp) by two
Subtract floor (comp/2) from the lower pixel value of the block
Add ceil (comp/2) to the upper pixel value of the block
The Stego-Image

Figure (3. 2): The Flow Chart of the PVD Method

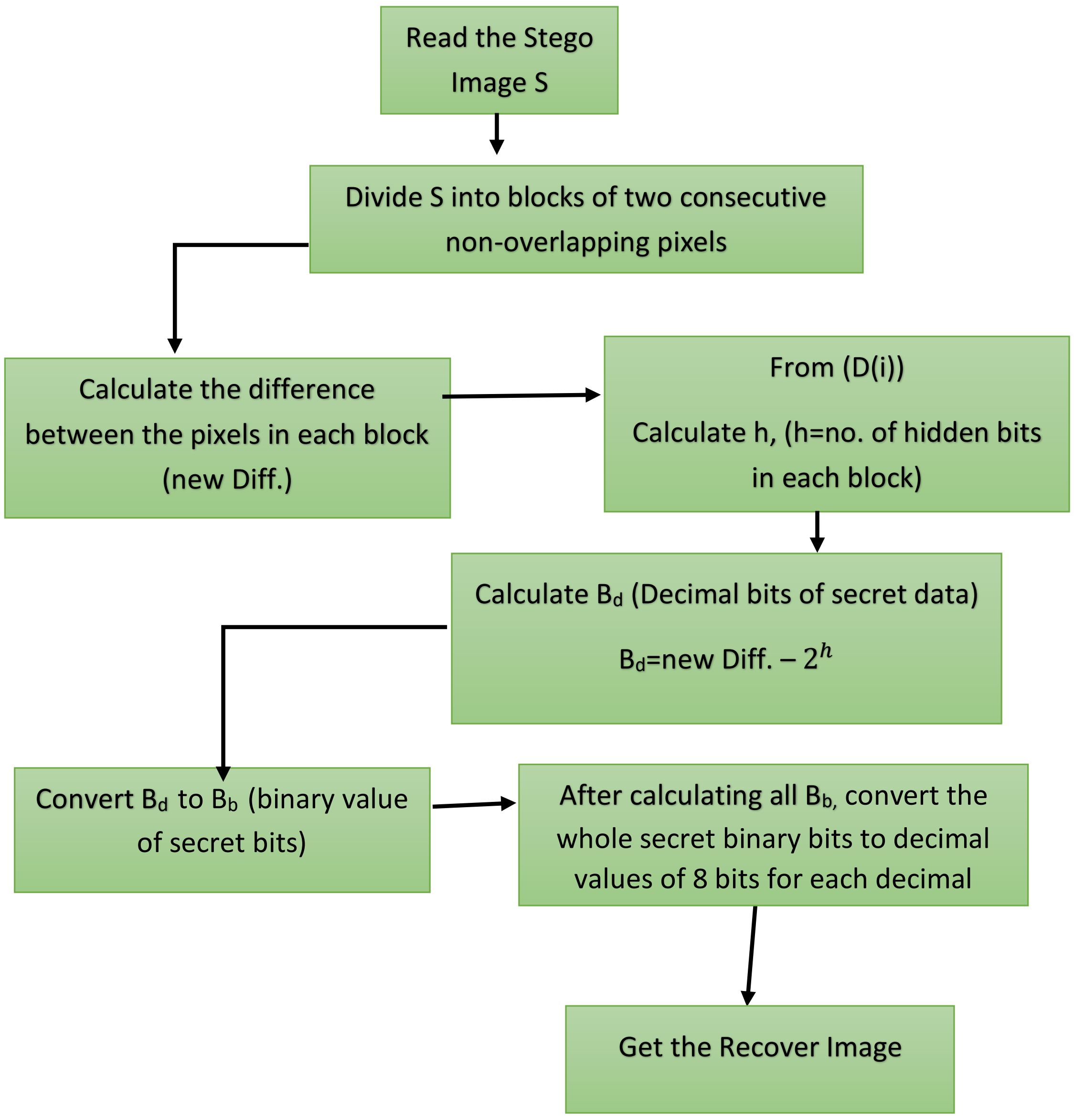

Read the Stego Image S
Divide S into blocks of two consecutive non-overlapping pixels
Calculate the difference between the pixels in each block (new Diff.)
From (D(i))
Calculate h, (h=no. of hidden bits in each block)
Calculate B_d (Decimal bits of secret data)
B_d=new Diff. – 2^h
Convert B_d to B_b (binary value of secret bits)
After calculating all B_b, convert the whole secret binary bits to decimal values of 8 bits for each decimal
Get the Recover Image

Figure (3. 3): The Reover Flow Chart of the PVD Method

Example:

Here, the gray values of a sample of two-pixel block are assumed to be (90; 110). The difference value is 20, which is in the range of 16 through 32. The width of the range is 16=2^4, which means that a difference value in the range can be used to ***embed four bits of secret data***. Assume that the four leading bits of the secret data are 1100. The value of this bit stream is 12 in Decimal. It is added to the lower bound value 16 of the range to yield the new difference value 28. Finally, the values (86; 114) are computed to be used as the gray values in the *stego image*. Note that 114-86=28.

3.6 Comparing LSB and PVD techniques by examples:

Reading the cover image C:

```
C = [23    30    42   47    41    33   30   50
     117  183  204  163  105   51   30   88
     100   59   17   26   16    17   14   18
     12    17   22   29   35    34   28   24
     25    38   50   90   194   237  166  70
     33    71   73   42   12    17   23   17
     19    11   16   21   32    21   22   22
     22    23   24   25   38    60   107  205];
```

1) ** First, we divide the image into non-overlapping blocks of two pixels in each block.

   ** Then, we calculate d_i (the difference value of the two pixels in each block) as follows:

   <u>Row 1:</u>

   d1 = |23-30| = 7

   d2 = |42-47| = 5

   d3 = |41-33| = 8

   d4 = |30-50| = 20

<u>Row 2:</u>

d1 = |117-183| = 66

d2 = |204-163| = 41

continuing …

.

.

<u>Row 8:</u>

d1 = |22-23| = 1

d2 = |24-25| = 1

d3 = |38-60| = 22

d4 = |107-205| = 98

The Computed Difference values are = {7 ,5 ,8 ,20 ,66 ,41 ,54 ,58 ,41 ,9 ,1 ,4 ,5 ,7 ,1 ,4 ,13 ,40 ,43 ,96 ,38 ,31 ,5 ,6 ,8 ,5 ,11 ,0 ,1 ,1 ,22 ,98 },**respectively.**

Now, we shall read the message to be hidden (secret message):

Suppose we try to hide the string name "Rana",

```
 'Rana' = [ 0    1    0    1    0    0    1    0
            0    1    1    0    0    0    0    1
            0    1    1    0    1    1    1    0
            0    1    1    0    0    0    0    1]
```

2) Determining Number of bits to be hidden :

According to the value of the difference between two adjacent pixels, we can determine the number of bits to hide in the *cover image* which can be calculated from the length of the interval where the pixels' difference belongs to, as shown in the following Figure [9] [12].

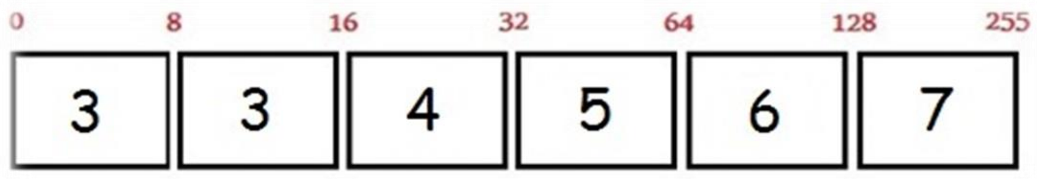


Figure (3. 4): No. of Bits to Hide in PVD Method

- First block : (23;30) -> d = 7 € Range [0,8] -> not taken (Because the new difference will be less than the old one, and this will give an unbalanced differences between the two pixels before and after modifying).

  So, we take the Range [8, 16].

  The width of the range = 8 $= 2^3$ → no. of bits to be hidden = 3.

  We take the first 3 bits from the secret message = $(0\ 1\ 0)_2 = (2)_{10}$

  Adding the lowest value of the interval to the value of the part of the secret message will give:

  i.e.: 8+2 = 10 -> (this is the new difference value)

  The Old difference =7

  Now we calculate (The new difference - the Old Difference):

  Diff. = 10-7=3.

  Divide the calculated value by two:

  (3/2) = 1.5

  Subtract [the lower approximation of the fraction = (1)] from the lower bound: 23 – 1 = 22

  Add [the upper approximation of the fraction = (2)] to the upper bound: 30 + 2 = 32

  So, the new pixel values = (22; 32)

  Note that 32-22=10 which equals the new difference value.

- block 2 : (42;47) -> d = 5 € Range [0,8] -> (not taken)

  We take the Range [8, 16].

  The width of the range = 8 = 2^3

  We take the next 3 bits from the secret message = $(1\ 0\ 0)_2$=$(4)_{10}$

  Adding the lowest value of the interval to the value of the part of the secret message:

  i.e.: 8+4 = 12 -> (this is the new difference value)

  Old difference =5

  So, the new pixel values = (39; 51)

  With the new difference = 12

- block 3 : (33;41) -> d = 8 € Range [8,16] -> taken

  The width of the range = 8 = 2^3

  We take the next 3 bits from the secret message = $(1\ 0\ 0)_2$ =$(4)_{10}$

  Adding the lowest value of the interval to the value of the part of the secret message:

  i.e.: 8+4 = 12 -> (this is the new difference value)

  Old difference =8

  So, the new pixel values = (31; 43)

  With the new difference = 12

- block 4 : (50;30) -> d = 20 € Range [16,32] -> (taken)

  The width of the range = 16 = 2^4

  We take the next 4 bits from the secret message = $(1\ 1\ 0\ 0)_2$ = $(12)_{10}$

  Adding the lowest value of the interval to the value of the part of the secret message:

i.e.: 16+12 = 28 -> (this is the new difference value)

Old difference = 20

So, the new pixel values = (26; 54)

With the new difference = 28.

- block 5 : (117;183) -> d = 66 € Range [64,128] -> (taken)

  The width of the range = 64 = 2^6

  We take the next 6 bits from the secret message = $(0\ 0\ 1\ 0\ 1\ 1)_2$ = $(11)_{10}$

  Adding the lowest value of the interval with to value of the part of the secret message:

  i.e.: 64+11 = 75 -> (this is the new difference value)

  Old difference =66

  So, the new pixel values = (113; 188)

  With the new difference = 75.

- block 6 : (204;163) -> d = 41 € Range [32,64] -> (taken)

  The width of the range = 32 = 2^5

  We take the next 5 bits from the secret message = $(0\ 1\ 1\ 1\ 0)_2$ = $(14)_{10}$

  Adding the lowest value of the interval to the value of the part

  of the secret message:

  i.e.: 32+14 = 46 -> (this is the new difference value)

  Old difference = 41

  So, the new pixel values = (207; 161)

  With the new difference = 46

- block 7 : (105;51) -> d = 54 € Range [32,64] -> (taken)

The width of the range = 32 = 2^5

We take the next 5 bits from the secret message = $(0\ 1\ 1\ 0\ 0)_2 = (12)_{10}$

Adding the lowest value of the interval to the value of the part

of the secret message:

i.e.: 32+12 = 44 -> (this is the new difference value)

Old difference = 54

So, the new pixel values = (116; 40)

With the new difference = 44.

- block 8 : (30;88) -> d = 58 € Range [32,64] -> (not taken)

  We take the Range [64,128].

  The width of the range = 64 = 2^6

  The remaining bits of the secret message are = $(0\ 0\ 1)_2 = (1)_{10}$

  Adding the lowest value of the interval to the value of the part

  of the secret message:

  i.e.: 64+1 = 65 -> (this is the new difference value)

  Old difference = 58

  So, the new pixel values = (27; 92)

  With the new difference = 58

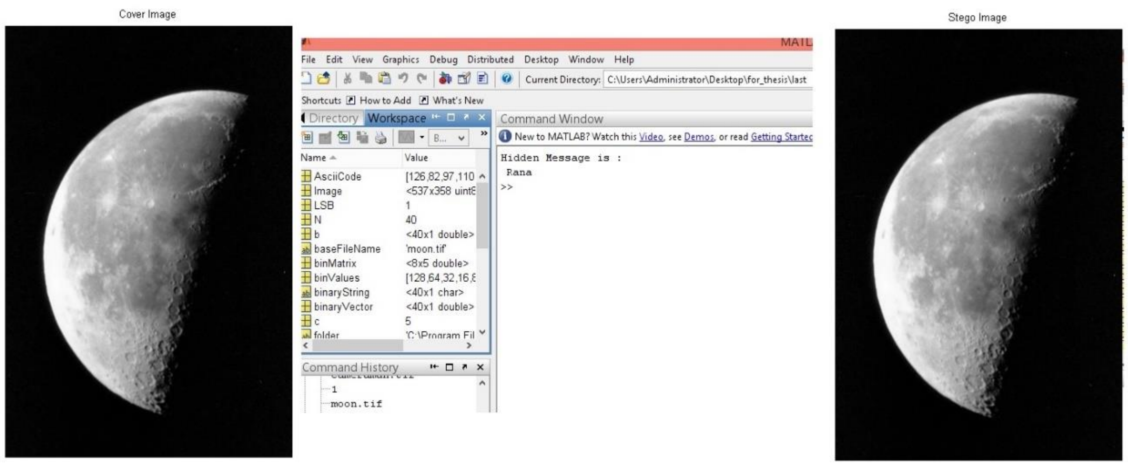
So, the New Pixel Values after embedding the secret message: (i.e.: the Cover image after embedding the secret message)

S=[22 32 39 51 31 43 26 54

113	**188**	**207**	**161**	**116**	**40**	**27**	**92**
100	59	17	26	16	17	14	18
12	17	22	29	35	34	28	24
25	38	50	90	194	237	166	70
33	71	73	42	12	17	23	17
19	11	16	21	32	21	22	22
22	23	24	25	38	60	107	205]

- ***The values of the pixels in the first two rows have been changed.***
- ***The values of the pixels in the last six rows remain the same.***

Using **Matlab**, we will never notice the change in the image relating to the original one after embedding process, as shown in the figure below.


Cover Image
MATL
File Edit View Graphics Debug Distributed Desktop Window Help
Current Directory: C:\Users\Administrator\Desktop\for_thesis\last
Shortcuts How to Add What's New
Directory Workspace
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Startec
Command History
Stego Image

Name	Value
AsciiCode	[126,82,97,110
Image	<537x358 uint8
LSB	1
N	40
b	<40x1 double>
baseFileName	'moon.tif'
binMatrix	<8x5 double>
binValues	[128,64,32,16,8
binaryString	<40x1 char>
binaryVector	<40x1 double>
c	5
folder	'C:\Program Fil

```
Hidden Message is :
 Rana
>>
```

```
1
moon.tif
```

Figure (3. 5): The Result of using PVD method in hiding text in an image

The same example Using Simple LSB Substitution:

Each pixel will embed only one bit of the secret message. First, we decide the number of bits to be embedded by reading the secret message.

In our example, the secret message:

‘Rana’ = [0 1 0 1 0 0 1 0

0 1 1 0 0 0 0 1

0 1 1 0 1 1 1 0

0 1 1 0 0 0 0 1]

4*8 = 32 Bit -→ So, number of pixels needed = 32 pixel to embed the whole secret message, **(i.e.: less capacity than using PVD method).**

Reading the cover image:

C = [23 30 42 47 41 33 30 50

117 183 204 163 105 51 30 88

100 59 17 26 16 17 14 18

12 17 22 29 35 34 28 24

25 38 50 90 194 237 166 70

33 71 73 42 12 17 23 17

19 11 16 21 32 21 22 22

22 23 24 25 38 60 107 205];

<u>Pixel 1:</u>

(23) = (0 0 0 1 0 1 1 1) → (0 0 0 1 0 1 1 0) = (22)

<u>Pixel 2:</u>

(30) = (0 0 0 1 1 1 1 0) → (0 0 0 1 1 1 1 1) = (31)

<u>Pixel 3:</u>

(42) = (0 0 1 0 1 0 1 0) → (0 0 1 0 1 0 1 0) = (42)

<u>Pixel 4:</u>

(47) = (0 0 1 0 1 1 1 1) → (0 0 1 0 1 1 1 1) = (47)

Continue pixel by pixel ….

.

.

.

<u>Pixel 32:</u>

(24) = (0 0 0 1 1 0 0 0) → (0 0 0 1 1 0 0 1) = (25)

So, The Cover Image before embedding process:

C = [23 30 42 47 41 33 30 50
117 183 204 163 105 51 30 88
100 59 17 26 16 17 14 18
12 17 22 29 35 34 28 24]

The Cover Image After embedding process (Stego-image):

S = [**22** 30 42 47 **40** **32** **31** 50
116 183 **205** **162** **104** **50** 30 **89**
100 59 17 26 **17** 17 **15** 18
12 17 **23** **28** **34** 34 28 **25**]

After embedding and using Simple LSB.:

We notice that the changes in pixel values occur in about half of the used pixels to embed the secret message.

i.e.: (17) pixel have been changed.

- ✓ Odd becomes Even → (1 -> 0)
- ✓ Even changes to Odd → (0 -> 1)

Comparison Results:

Table (3. 1): Comparison Results of Simple LSB and PVD steganographic techniques

Method	**MSE**	**PSNR**	**Capacity**
Simple LSB	0.5	51.1411	Low
PVD	9.1094	38.5359	High

**** According to the above results, we note that PVD method provides higher capacity than LSB substitution method.**

Examples:

Both cover image and secret image have the same size:

We read two images and check the size of both of them:

Cover image ***(rice.tif)***:

```
coverImage = imread('moon.tif');
% Get the number of rows and columns in the cover image.
[visibleRows visibleColumn] = size(coverImage);
```

Results are:

```
visibleRows = 256
visibleColumns = 256
```

Image to be hidden ***(Cameraman.tif)***:

```
hiddenImage = imread('cameraman.tif');
% Get the number of rows and columns in the hidden image.
[hiddenRows hiddenColumns] = size(hiddenImage);
```

Results are:

```
hiddenRows = 256
```

```
hiddenColumns = 256
```

From the above results, note that the size of the hidden image is equal to the size of the cover image. The results of PVD method are shown in the following table:

Table (3. 2): Results of PVD method (Both size of the cover image and the hidden image are equal)

Method	MSE	PSNR
PVD	2.0318	45.0520

Figure (3. 6): Results of PVD Method (both the cover and the hidden images have the same size)

3.7 Different Image Sizes - Matching the size of cover image and hidden image:

1) Tiling the Secret Image:

We read both cover and secret images and check the size of both of them:

Cover image ***(Moon.tif)***:

```
coverImage = imread('moon.tif');
```

% Get the number of rows and columns in the cover image.

[visibleRows visibleColumn] = size(coverImage);

Results are:

visibleRows = 537

visibleColumns = 358

Image to be hidden ***(Cameraman.tif)***:

hiddenImage = imread('cameraman.tif');

% Get the number of rows and columns in the hidden image.

[hiddenRows hiddenColumns] = size(hiddenImage);

Results are:

hiddenRows = 256

hiddenColumns = 256

From the above results, note that the size of the hidden image is smaller than that of the cover image; so, we need to tile the hidden image to match the size of the cover image. The results of PVD method are shown in the following table:

Table (3. 3): Results of PVD method (the size of the cover image is greater than the size of the hidden image)

Method	MSE	PSNR
PVD	6.8856	39.7514

Figure (3. 7): Results of PVD Method (with tiling the hidden image)

2) Scaling down the secret image:

We read both cover and secret images and check the size of both of them:

Cover image ***(cell.tif)***:

```
coverImage = imread('cell.tif');
% Get the number of rows and columns in the cover image.
[visibleRows visibleColumn] = size(coverImage);
```

Results are:

```
visibleRows = 159
visibleColumns = 191
```

Image to be hidden ***(coins.png)***:

```
hiddenImage = imread('coins.png');
% Get the number of rows and columns in the hidden image.
[hiddenRows hiddenColumns] = size(hiddenImage);
```

Results are:

```
hiddenRows = 246
hiddenColumns = 300
```

We can notice that the size of the cover image is smaller than the size of the secret image, so we need to scale down the size of the secret image to match that of the cover image, then do the embedding process, the results of PVD method are shown as follows:

Table (3. 4): Results of PVD method (the size of the cover image is less than the size of the hidden image)

Method	MSE	PSNR
PVD	8.0049	39.0973

Figure (3. 8): Results of PVD Method (with scaling down the hidden image)

Chapter Four

Dynamic Least Significant Bit (LSB) Substitution Method

Chapter Four

Dynamic LSB

4.1 Goal of using Dynamic LSB:

To distribute the hidden data in the *cover image* according to the ratio between the size of the data to be hidden and the size of the *cover image*, so that to reduce the distortion in the *stego image* and improve the quality of the simple least significant bit substitution method.

4.2 Algorithm and Flow Chart of Dynamic LSB Substitution Method:

Dynamic LSB Algorithm:

Inputs: *Cover Image, Hidden message.*

Procedure:

```
Step1: Read the cover image (C) → (r=no. of rows, c=no. of columns, M=size of the cover image)
Step2: Read the Hidden message → (b=no. of bits to be hidden)
Step3: Calculate the Ratio $R = \frac{b}{M}$
Step4: If (R <= 1)
    For i = 1 to $\frac{b}{c}$
    Do LSB in row i
    i = i + ceil $\left(\frac{1}{R}\right)$
    End
```

Output: *Stego Image.*

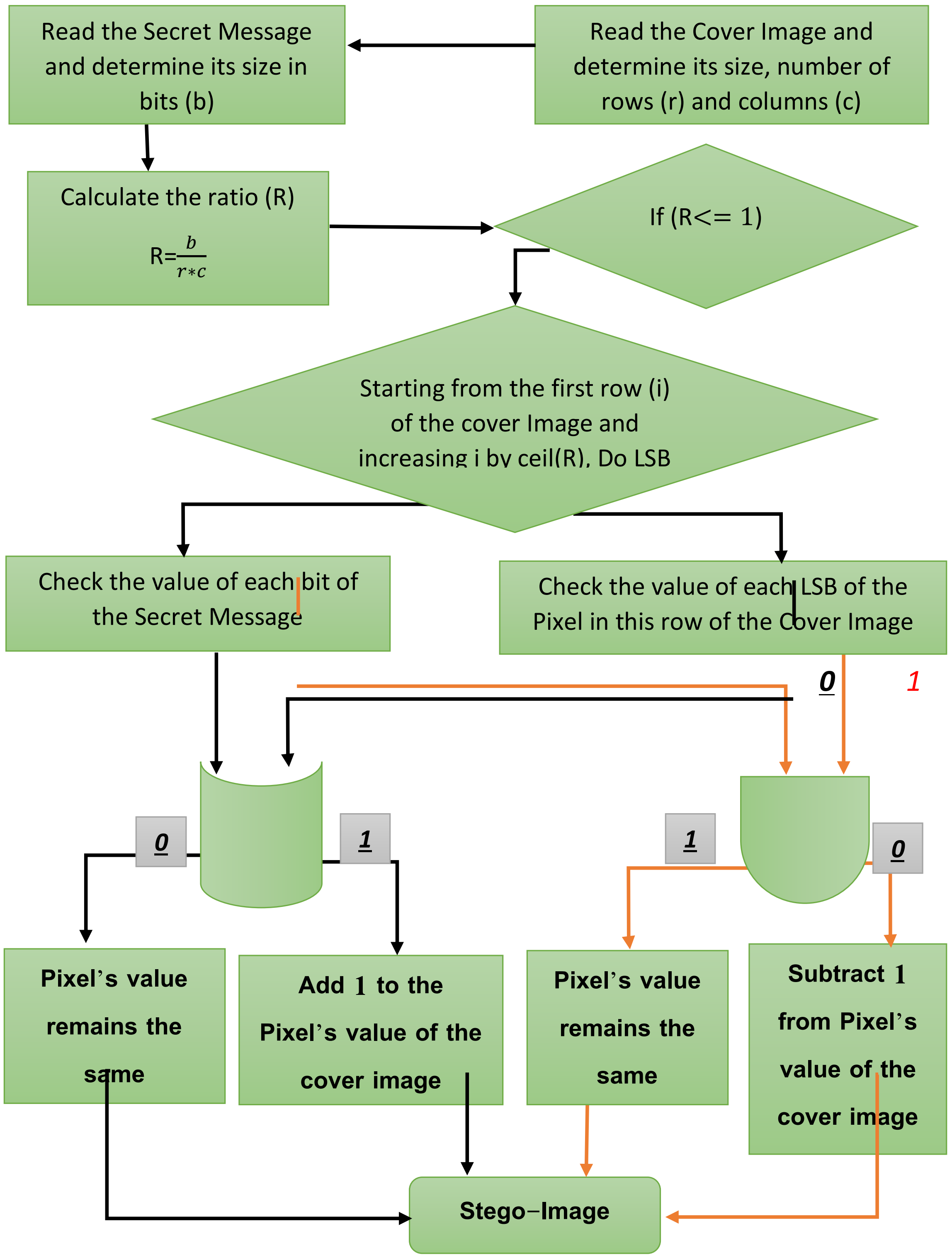

Read the Secret Message and determine its size in bits (b)
Read the Cover Image and determine its size, number of rows (r) and columns (c)
Calculate the ratio (R)
R=$\frac{b}{r*c}$
If (R<= 1)
Starting from the first row (i) of the cover Image and increasing i by ceil(R), Do LSB
Check the value of each bit of the Secret Message
Check the value of each LSB of the Pixel in this row of the Cover Image
0
1
0
1
1
0
Pixel's value remains the same
Add 1 to the Pixel's value of the cover image
Pixel's value remains the same
Subtract 1 from Pixel's value of the cover image
Stego–Image

Figure (4. 1): The Flow Chart of Dynamic LSB Substitution Method

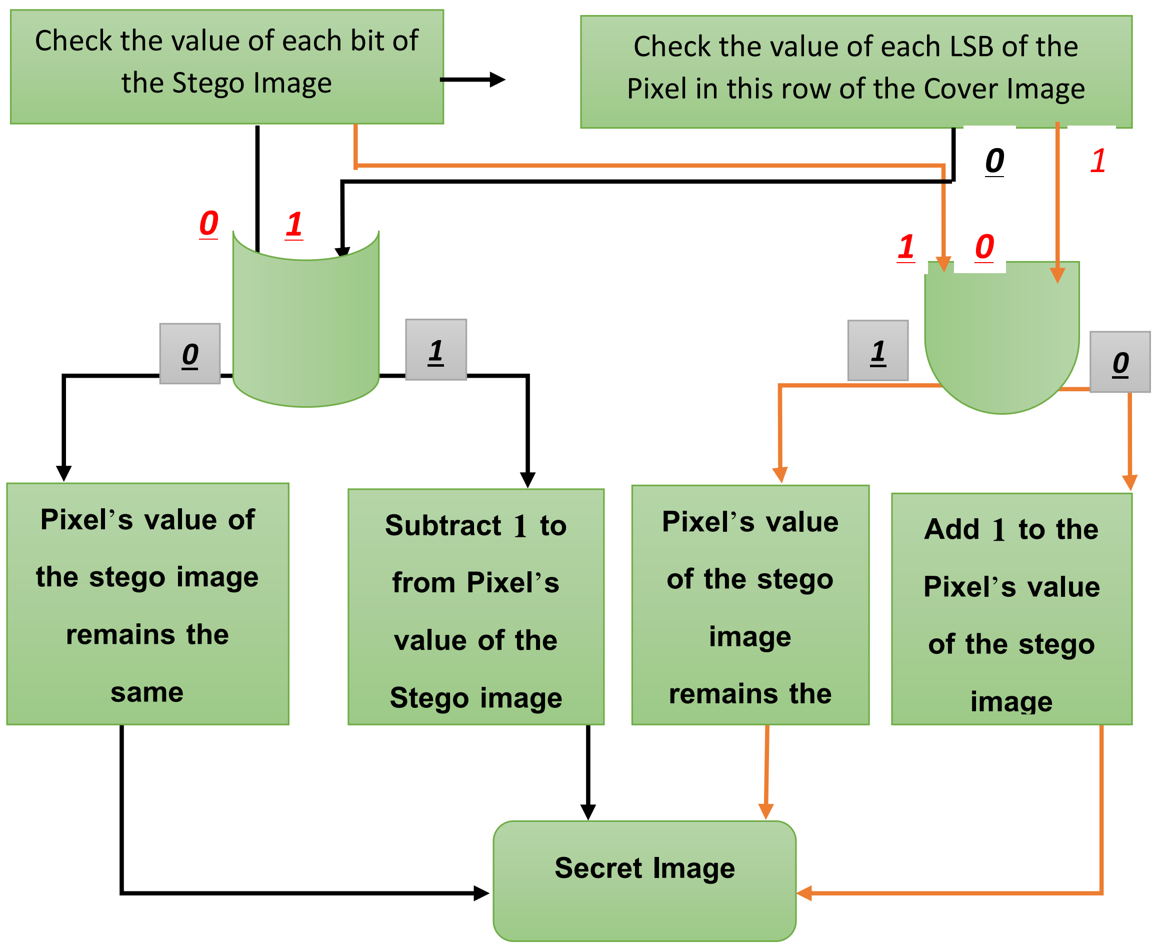

Check the value of each bit of the Stego Image
Check the value of each LSB of the Pixel in this row of the Cover Image
0
1
1
0
0
1
0
1
1
0
Pixel's value of the stego image remains the same
Subtract 1 to from Pixel's value of the Stego image
Pixel's value of the stego image remains the
Add 1 to the Pixel's value of the stego image
Secret Image

Figure (4. 2): The Recover Flow Chart of Dynamic LSB Substitution Method

4.3 Examples:

**** Taking the 'moon.tif' image as the cover image:***

r= number of rows of the cover image.

c= number of columns of the cover image.

b= number of bits of the secret message (message to be hidden).

M= size of the cover image.

Example1:

r=8, c=8 → M=64

b=32

Cover Image:

$$\mathbf{C} = \begin{bmatrix} 23 & 30 & 42 & 47 & 41 & 33 & 30 & 50 \\ 117 & 183 & 204 & 163 & 105 & 51 & 30 & 88 \\ 100 & 59 & 17 & 26 & 16 & 17 & 14 & 18 \\ 12 & 17 & 22 & 29 & 35 & 34 & 28 & 24 \\ 25 & 38 & 50 & 90 & 194 & 237 & 166 & 70 \\ 33 & 71 & 73 & 42 & 12 & 17 & 23 & 17 \\ 19 & 11 & 16 & 21 & 32 & 21 & 22 & 22 \\ 22 & 23 & 24 & 25 & 38 & 60 & 107 & 205 \end{bmatrix};$$

Secret Message:

$$\textbf{'Rana'} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

The ratio between the size of the data to be hidden and the size of the cover image is → (b: M = 32:64= 1:2).

*****So that, we can hide the message in row1, row3, row5... etc.***

Pixel 1:

$(23)_{10} = (0\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 1\ 1\ 0)_2 = (22)_{10}$

Pixel 2:

$(30)_{10} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 1\ 1\ 1\ 1)_2 = (31)_{10}$

Pixel 3:

$(42)_{10} = (0\ 0\ 1\ 0\ 1\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 0\ 1\ 0)_2 = (42)_{10}$

Pixel 4:

$(47)_{10} = (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 = (47)_{10}$

Pixel 5:

$(41)_{10} = (0\ 0\ 1\ 0\ 1\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 0\ 0\ 0)_2 = (40)_{10}$

Pixel 6:

$(33)_{10} = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)_2 = (32)_{10}$

Pixel 7:

$(30)_{10} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 = (31)_{10}$

Pixel 8:

$(50)_{10} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 = (50)_{10}$

Pixel 17:

$(100)_{10} = (0\ 1\ 1\ 0\ 0\ 1\ 0\ 0)_2 \rightarrow (0\ 1\ 1\ 0\ 0\ 1\ 0\ 0)_2 = (100)_{10}$

Pixel 18:

$(59)_{10} = (0\ 0\ 1\ 1\ 1\ 0\ 1\ 1)_2 \rightarrow (0\ 0\ 1\ 1\ 1\ 0\ 1\ 1)_2 = (59)_{10}$

<u>Pixel 19:</u>

$(17)_{10} = (0\ 0\ 0\ 1\ 0\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 0\ 0\ 1)_2 = (17)_{10}$

<u>Pixel 20:</u>

$(26)_{10} = (0\ 0\ 0\ 1\ 1\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 1\ 0\ 1\ 0)_2 = (26)_{10}$

<u>Pixel 21:</u>

$(16)_{10} = (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)_2 = (16)_{10}$

<u>Pixel 22:</u>

$(17)_{10} = (0\ 0\ 0\ 1\ 0\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)_2 = (16)_{10}$

<u>Pixel 23:</u>

$(14)_{10} = (0\ 0\ 0\ 0\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 0\ 1\ 1\ 1\ 0)_2 = (14)_{10}$

<u>Pixel 24:</u>

$(18)_{10} = (0\ 0\ 0\ 1\ 0\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 0\ 1\ 1)_2 = (19)_{10}$

<u>Pixel 33:</u>

$(25)_{10} = (0\ 0\ 0\ 1\ 1\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 0\ 1\ 1\ 0\ 0\ 0)_2 = (24)_{10}$

<u>Pixel 34:</u>

$(38)_{10} = (0\ 0\ 1\ 0\ 0\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 0\ 1\ 1\ 1)_2 = (39)_{10}$

<u>Pixel 35:</u>

$(50)_{10} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)_2 = (51)_{10}$

<u>Pixel 36:</u>

$(90)_{10} = (0\ 1\ 0\ 1\ 1\ 0\ 1\ 0)_2 \rightarrow (0\ 1\ 0\ 1\ 1\ 0\ 1\ 0)_2 = (90)_{10}$

<u>Pixel 37:</u>

$(194)_{10} = (1\ 1\ 0\ 0\ 0\ 0\ 1\ 0)_2 \rightarrow (1\ 1\ 0\ 0\ 0\ 0\ 1\ 1)_2 = (194)_{10}$

<u>Pixel 38:</u>

$(237)_{10} = (1\ 1\ 1\ 0\ 1\ 1\ 0\ 1)_2 \rightarrow (1\ 1\ 1\ 0\ 1\ 1\ 0\ 1)_2 = (237)_{10}$

<u>Pixel 39:</u>

$(166)_{10} = (1\ 0\ 1\ 0\ 0\ 1\ 1\ 0)_2 \rightarrow (1\ 0\ 1\ 0\ 0\ 1\ 1\ 1)_2 = (167)_{10}$

<u>Pixel 40:</u>

$(70)_{10} = (0\ 1\ 0\ 0\ 0\ 1\ 1\ 0)_2 \rightarrow (0\ 1\ 0\ 0\ 0\ 1\ 1\ 0)_2 = (70)_{10}$

<u>Pixel 49:</u>

$(19)_{10} = (0\ 0\ 0\ 1\ 0\ 0\ 1\ 1)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 0\ 1\ 0)_2 = (18)_{10}$

<u>Pixel 50:</u>

$(11)_{10} = (0\ 0\ 0\ 0\ 1\ 0\ 1\ 1)_2 \rightarrow (0\ 0\ 0\ 0\ 1\ 0\ 1\ 1)_2 = (11)_{10}$

<u>Pixel 51:</u>

$(16)_{10} = (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 0\ 0\ 1)_2 = (17)_{10}$

<u>Pixel 52:</u>

$(21)_{10} = (0\ 0\ 0\ 1\ 0\ 1\ 0\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 1\ 0\ 0)_2 = (21)_{10}$

<u>Pixel 53:</u>

$(32)_{10} = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)_2 = (32)_{10}$

<u>Pixel 54:</u>

$(21)_{10} = (0\ 0\ 0\ 1\ 0\ 1\ 0\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 1\ 0\ 0)_2 = (21)_{10}$

<u>Pixel 55:</u>

$(22)_{10} = (0\ 0\ 0\ 1\ 0\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 1\ 1\ 0)_2 = (22)_{10}$

<u>Pixel 56:</u>

$(22)_{10} = (0\ 0\ 0\ 1\ 0\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_2 = (23)_{10}$

So that, the resulting *stag image*:

S= **[22 31 42 47 40 32 31 50**

117 183 204 163 105 51 30 88

100 59 17 26 16 16 14 19

12 17 22 29 35 34 28 24

24 39 51 90 194 237 167 70

33 71 73 42 12 17 23 17

18 11 17 21 32 21 22 23

22 23 24 25 38 60 107 205]

Example2:

If b=16 →b:M =16:64 = 1:4

Using the same previous cover image with the Secret message:

'Ra' = [0 1 0 1 0 0 1 0

0 1 1 0 0 0 0 1]

<u>Pixel 1:</u>

$(23)_{10} = (0\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 1\ 1\ 0)_2 = (22)_{10}$

<u>Pixel 2:</u>

$(30)_{10} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 1\ 1\ 1\ 1)_2 = (31)_{10}$

<u>Pixel 3:</u>

$(42)_{10} = (0\ 0\ 1\ 0\ 1\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 0\ 1\ 0)_2 = (42)_{10}$

<u>Pixel 4:</u>

$(47)_{10} = (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 = (47)_{10}$

Pixel 5:

$(41)_{10} = (0\ 0\ 1\ 0\ 1\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 0\ 0\ 0)_2 = (40)_{10}$

Pixel 6:

$(33)_{10} = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)_2 = (32)_{10}$

Pixel 7:

$(30)_{10} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 = (31)_{10}$

Pixel 8:

$(50)_{10} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 = (50)_{10}$

Pixel 33:

$(25)_{10} = (0\ 0\ 0\ 1\ 1\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 0\ 1\ 1\ 0\ 0\ 0)_2 = (24)_{10}$

Pixel 34:

$(38)_{10} = (0\ 0\ 1\ 0\ 0\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 0\ 1\ 1\ 1)_2 = (39)_{10}$

Pixel 35:

$(50)_{10} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)_2 = (51)_{10}$

Pixel 36:

$(90)_{10} = (0\ 1\ 0\ 1\ 1\ 0\ 1\ 0)_2 \rightarrow (0\ 1\ 0\ 1\ 1\ 0\ 1\ 0)_2 = (90)_{10}$

Pixel 37:

$(194)_{10} = (1\ 1\ 0\ 0\ 0\ 0\ 1\ 0)_2 \rightarrow (1\ 1\ 0\ 0\ 0\ 0\ 1\ 0)_2 = (194)_{10}$

Pixel 38:

$(237)_{10} = (1\ 1\ 1\ 0\ 1\ 1\ 0\ 1)_2 \rightarrow (1\ 1\ 1\ 0\ 1\ 1\ 0\ 0)_2 = (236)_{10}$

Pixel 39:

$(166)_{10} = (1\ 0\ 1\ 0\ 0\ 1\ 1\ 0)_2 \rightarrow (1\ 0\ 1\ 0\ 0\ 1\ 1\ 0)_2 = (166)_{10}$

Pixel 40:

$(70)_{10} = (0\ 1\ 0\ 0\ 0\ 1\ 1\ 0)_2 \rightarrow (0\ 1\ 0\ 0\ 0\ 1\ 1\ 1)_2 = (71)_{10}$

So that, the resulting *stego image*:

S= [22 31 42 47 40 32 31 50

117 183 204 163 105 51 30 88

100 59 17 26 16 17 14 18

12 17 22 29 35 34 28 24

24 39 51 90 194 236 166 71

33 71 73 42 12 17 23 17

19 11 16 21 32 21 22 22

22 23 24 25 38 60 107 205];

Example3:

If b=8 $\rightarrow$b:M = 8:64 = 1:8

Using the same previous cover image with the Secret message:

'R' = [0 1 0 1 0 0 1 0]

<u>Pixel 1:</u>

$(23)_{10} = (0\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 1\ 1\ 0)_2 = (22)_{10}$

<u>Pixel 2:</u>

$(30)_{10} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 1\ 1\ 1\ 1)_2 = (31)_{10}$

<u>Pixel 3:</u>

$(42)_{10} = (0\ 0\ 1\ 0\ 1\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 0\ 1\ 0)_2 = (42)_{10}$

<u>Pixel 4:</u>

$(47)_{10} = (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 = (47)_{10}$

<u>Pixel 5:</u>

$(41)_{10} = (0\ 0\ 1\ 0\ 1\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 0\ 0\ 0)_2 = (40)_{10}$

<u>Pixel 6:</u>

$(33)_{10} = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)_2 = (32)_{10}$

<u>Pixel 7:</u>

$(30)_{10} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 = (31)_{10}$

<u>Pixel 8:</u>

$(50)_{10} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 = (50)_{10}$

So that, the resulting stego image:

S= [**22** **31** 42 47 **40** **32** **31** 50

117 183 204 163 105 51 30 88

100 59 17 26 16 17 14 18

12 17 22 29 35 34 28 24

25 38 50 90 194 237 166 70

33 71 73 42 12 17 23 17

19 11 16 21 32 21 22 22

22 23 24 25 38 60 107 205];

Example4:

b=16, M= 24

b:M = 16:24 = 2:3

C = [23 30 42 47 41 33 30 50

117 183 204 163 105 51 30 88

100 59 17 26 16 17 14 18]

The secret message to be hidden:

‘Ra’ = [0 1 0 1 0 0 1 0

0 1 1 0 0 0 0 1].

<u>Pixel 1:</u>

$(23)_{10} = (0\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_2 \rightarrow (0\ 0\ 0\ 1\ 0\ 1\ 1\ 0)_2 = (22)_{10}$

<u>Pixel 2:</u>

$(30)_{10} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 1\ 1\ 1\ 1)_2 = (31)_{10}$

<u>Pixel 3:</u>

$(42)_{10} = (0\ 0\ 1\ 0\ 1\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 0\ 1\ 0)_2 = (42)_{10}$

<u>Pixel 4:</u>

$(47)_{10} = (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 = (47)_{10}$

<u>Pixel 5:</u>

$(41)_{10} = (0\ 0\ 1\ 0\ 1\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 0\ 0\ 0)_2 = (40)_{10}$

<u>Pixel 6:</u>

$(33)_{10} = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 1)_2 \rightarrow (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)_2 = (32)_{10}$

<u>Pixel 7:</u>

$(30)_{10} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2 = (31)_{10}$

<u>Pixel 8:</u>

$(50)_{10} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 \rightarrow (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 = (50)_{10}$

<u>Pixel 9:</u>

$(117)_{10} = (0\ 1\ 1\ 1\ 0\ 1\ 0\ 1)_2 \rightarrow (0\ 1\ 1\ 1\ 0\ 1\ 0\ 0)_2 = (116)_{10}$

<u>Pixel 10:</u>

$(183)_{10} = (1\ 0\ 1\ 1\ 0\ 1\ 1\ 1)_2 \rightarrow (1\ 0\ 1\ 1\ 0\ 1\ 1\ 1)_2 = (183)_{10}$

<u>Pixel 11:</u>

$(204)_{10} = (1\ 1\ 0\ 0\ 1\ 1\ 0\ 0)_2 \rightarrow (1\ 1\ 0\ 0\ 1\ 1\ 0\ 1)_2 = (205)_{10}$

Pixel 12:

$(163)_{10} = (1\ 0\ 1\ 0\ 0\ 0\ 1\ 1)_2 \rightarrow (1\ 0\ 1\ 0\ 0\ 0\ 1\ 0)_2 = (162)_{10}$

Pixel 13:

$(105)_{10} = (0\ 1\ 1\ 0\ 1\ 0\ 0\ 1)_2 \rightarrow (0\ 1\ 1\ 0\ 1\ 0\ 0\ 0)_2 = (104)_{10}$

Pixel 14:

$(51)_{10} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)_2 \rightarrow (0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2 = (50)_{10}$

Pixel 15:

$(30)_{10} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 \rightarrow (0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)_2 = (30)_{10}$

Pixel 16:

$(88)_{10} = (0\ 1\ 0\ 1\ 1\ 0\ 0\ 0)_2 \rightarrow (0\ 1\ 0\ 1\ 1\ 0\ 0\ 1)_2 = (89)_{10}$

```
S = [22   31   42   47   40   32   31  50
     116  183  205  162  104  50  30  89
     100   59   17   26   16  17  14  18];
```

4.4 Results:

Table (4. 1): Results of different cases of Dynamic LSB

Ratio between Cover and Stego Images	MSE	PSNR
1:2	0.2188	54.713
1:4	0.1563	56.1926
1:8	0.0781	59.2029
2:3	0.4583	51.519

- The lower the value of MSE, the better the quality of *stego image*.
- The higher the value of PSNR, the better the quality of *stego image*.

Stego Images in each case:

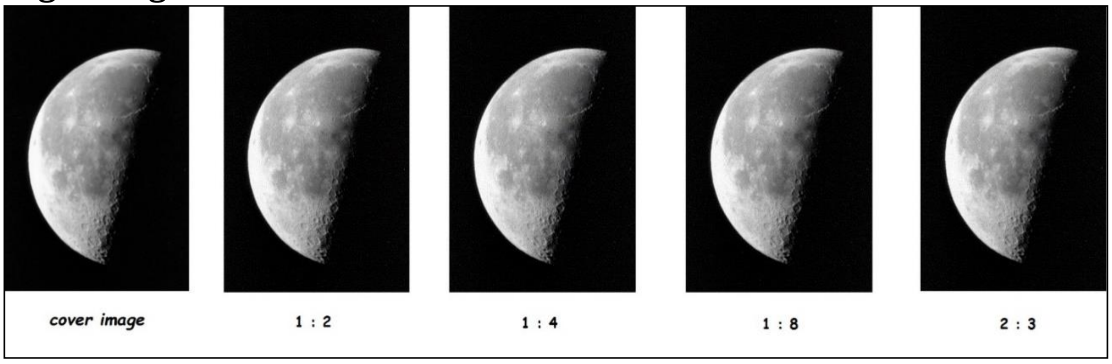


Fig.20: Results of different cases of Dynamic LSB

Conclusion:

In various Hiding methods such as LSB, PVD and dynamic LSB, one of the most important things is that the quality of the stego-image must not be degraded too much after embedding to get the goal of those data hiding methods which is to make the stego-image indistinguishable to grabbers. Thus, if the process of embedding degrades the quality of the stego-image too much, any grabber will easily take notice of it.

Using Pixel Value Differencing method (PVD), the quality of the image combined with the capacity of hidden data is improved, as well as the complexity and robustness of embedded data are increased where the secret data is stored in a difficult way to understand by any intruder.

In our thesis, data hiding schemes are used, by applying some pixel adjustment process to the stego-image obtained by the simple LSB substitution method, and using PVD method, the image quality of the stego-image can be greatly improved with low extra computational complexity. The mean-square- error between the stego-image and the cover-image and

Peak Signal to noise ratio are computed. Experimental results show that the stego-image is visually indistinguishable from the original cover-image.

Summary of results of various hiding information methods:

**** Note: the value (1) is given to the best method comparing with others.**

Table (4. 2):Comparative Results of Different Hiding Information Techniques

Method	Implementation	Image Quality	Capacity Of Secret Data
Simple LSB	1	3	4
LSB (2,3,..,8)	2	4	2
Dynamic LSB	3	1	3
PVD	4	2	1

- According to the above results, we find that ***PVD*** is the best method in hiding capacity, where ***Dynamic LSB*** is the best in Image quality.

References

1. Sabu M Thampi, "**Information Hiding Techniques: A Tutorial Review**", Department of Computer Science & Engineering LBS College of Engineering, Kasaragod Kerala- 671542, S.India.
2. Muhalim Mohamed Amin,Subariah Ibrahim, Mazleena Salleh Mohd, Rozi Katmin, "**Information Hiding Using Steganography**", Department of Computer System & Communication Faculty of Computer Science and Information system ,University Technology Malaysia.
3. Philip Bateman, "**Image Steganography and Steganalysis**", Department of Computing, Faculty of Engineering and Physical Sciences, University of Surrey.
4. R. Amirtharajan, R. Akila, P. Deepikachowdavarapu," ***A Comparative Analysis of Image Steganography*", International Journal of Computer Applications** (0975 – 8887), Volume 2 – No. 3, May 2010
5. Enhanced Data Hiding Capacity Using LSB-Based Image Steganography Method.
6. Aishwary Kulshreshta , Ankur Goyal,"***Image Steganography Using Dynamic LSB with Blowfish Algorithm***", **International Journal of Computer & Organization Trends** –Volume 3 Issue 7 – August 2013
7. X. Liao, Q.-Y. Wen, and J. Zhang, "***A steganographic method for digital images with four-pixel differencing and modified LSB substitution***," **Journal of Visual Communication and Image Representation**, vol. 22, no. 1, pp. 1–8, 2011.

8. Marghny H. Mohamed1, Naziha M. AL-Aidroos2, and Mohamed A. Bamatraf3," ***A Combined Image Steganography Technique Based on Edge Concept & Dynamic LSB***", **International Journal of Engineering Research & Technology (IJERT)** Vol. 1 Issue 8, October - 2012 ISSN: 2278-0181
9. Jagruti Salunkhe and Sumedha Sirsikar , "***Pixel Value Differencing a Steganographic method: A Survey"*** **International Journal of Computer Applications (0975 –8887) International Conference on Recent Trends in engineering & Technology** - 2013(ICRTET'2013).
10. Wu,Tsai, "***A steganographic method for images by pixel-value differencing***" ,Volume 24, Issues 9-10, June 2003, pages 1613-1626
11. Hsien-Wen Tseng1 and Hui-Shih Leng,"**A Steganographic Method Based on Pixel-Value Differencing and the Perfect Square Number**"
12. D.-C. Wu and W.-H. Tsai, "***A steganographic method for images by pixel-value differencing***," **Pattern Recognition Letters**, vol. 24, no. 9-10, pp. 1613–1626, 2003. View at Publisher · View at Google Scholar · View at Scopus.
13. H. C. Wu, N. I. Wu, C. S. Tsai, and M. S. Huang, "***Image steganographic scheme based on pixel-value differencing and LSB replacement methods,"*** **IEE Proceedings Vision Image and Signal Processing**, vol. 152, no. 5, pp. 611–615, 2005. View at Google Scholar0.
14. C. H. Yang and C. Y. Weng, "***A steganographic method for digital images by multi-pixel differencing,***" in ***Proceedings of International Computer Symposium,*** pp. 831–836, Taipei, Taiwan, 2006.

15. K.-H. Jung, K.-J. Ha, and K.-Y. Yoo***, "Image data hiding method based on multi-pixel differencing and LSB substitution methods,"*** **in Proceedings of International Conference on Convergence and Hybrid Information Technology (ICHIT '08),** pp. 355–358, kor, August 2008. View at Publisher · View at Google Scholar · View at Scopus

16. J.-C. Liu and M.-H. Shih, ***"Generalizations of pixel-value differencing steganography for data hiding in images,"*** **Fundamenta Informaticae,** vol. 83, no. 3, pp. 319–335, 2008. View at Google Scholar · View at Zentralblatt MATH · View at MathSciNet · View at Scopus.

17. C.-H. Yang, C.-Y. Weng, H.-K. Tso, and S.-J. Wang, ***"A data hiding scheme using the varieties of pixel-value differencing in multimedia images,"*** **Journal of Systems and Software,** vol. 84, no. 4, pp. 669–678, 2011. View at Publisher · View at Google Scholar · View at Scopus.

18. C. H. Yang, S. J. Wang, C. Y. Weng, and H. M. Sun, ***"Information hiding technique based on blocked PVD,"*** **Journal of Information Management,** vol. 15, no. 3, pp. 29–48, 2008. View at Google Scholar.

19. Arvind Kumar, Km. Pooja, ***"Steganography- A Data Hiding Technique"*** **International Journal of Computer Applications ISSN** 0975– 8887, Volume 9– No.7, November 2010.

20. Muhalim Mohamed Amin, Saurabh Ibrahim, Mazleena Salleh, Mohd. Rozi Katmin, "***Information hiding using steganography***",Vol71847 ,2003.

21. Masoud Afrakhteh, Subariah Ibrahim *"Adaptive steganography scheme Using More Surrounding Pixels"*, **International Conference On Computer Design And Applications (ICCDA 2010)**, Vol. 1, V1225-V1229.

22. Wu,Tsai, "***A steganographic method for images by pixel-value differencing"*** ,Volume 24, Issues 9-10, June 2003, pages 1613-1626.

23. H. C. Wu, Tsai, Hwang, ***"Image steganographic scheme based on pixel-value differencing and LSB replacement methods"***, **IEE Proc-vis**. Image signal process, vol. 152, no. 5, October 2005, 611-615.

24. Han-ling ZHANG, Guang-zhi GENG, Cai-qiong Xing, 2009. "***Image Steganography using Pixel-Value Differencing"***, **IEEE DOI** 10. 1109/ISECS. 2009. 139), 109–112.

25. H. B. Kekre, Archana Athawale, Pallavi N. Halarnkar, "***Performance Evaluation of Pixel Value Differencing and Kekre's Modified Algorithm for Information Hiding in Images",*** **International Conference on Advances in Computing,** Communication and Control (ICAC3'09).

26. A. L. Khade. B. G. Hogde, V B Gaikwad. "***Secret Communication via Image Hiding In Image by Pixel Value Differencing***", ICWET', February 2010, 437-438.

Appendix

Appendix:

Matlab code to compute MSE and PSNR:

```
origImg = double(originalImage);   %cover image
distImg = double(watermarkedImage);   %stego image


[M N] = size(origImg);
distImg1=imresize(distImg,[M N]);
error = origImg - distImg1;
MSE = sum(sum(error .* error)) / (M * N)
if(MSE > 0)
      PSNR = 10*log10((255^2)/MSE)
else
   PSNR =0;
end
```

Matlab Code to scale down the hidden image:

```
if hiddenRows > visibleRows || hiddenColumns > visibleColumns
   amountToShrink = min([visibleRows / hiddenRows, visibleColumns / hiddenColumns]);
   binaryImage = imresize(binaryImage, amountToShrink);
   % Need to update the number of rows and columns.
   [hiddenRows hiddenColumns] = size(binaryImage);
End
```

Matlab Code to tile the hidden image:

```
if hiddenRows < visibleRows || hiddenColumns < visibleColumns
   watermark = zeros(size(originalImage), 'uint8');
   for column = 1:visibleColumns
      for row = 1:visibleRows
         watermark(row, column) = binaryImage(mod(row,hiddenRows)+1,
mod(column,hiddenColumns)+1);
      end
   end
```

Matlab code for LSB substitution method:

```
Image = imread(fullFileName);
msg='Rana';
c(1:1:1)= length(msg) ;
c=imresize(c,[size(c,1) size(c,2)],'nearest');
message = msg
message = strtrim(message);
m = length(message) * 8;
AsciiCode = uint8(message);
binaryString = transpose(dec2bin(AsciiCode,8));
binaryString = binaryString(:);
N = length(binaryString);
b = zeros(N,1);
for k = 1:N
```

```
  if(binaryString(k) == '1')
      b(k) = 1;
  else
      b(k) = 0;
  end
end
s = c;
  height = size(c,1);
  width = size(c,2);
k = 1;
for i = 1 : height
  for j = 1 : width
      LSB = mod(double(c(i,j)), 2);
      if (k>m || LSB == b(k))
          s(i,j) = c(i,j);
      elseif(LSB == 1)
          s(i,j) = (c(i,j) - 1);
      elseif(LSB == 0)
          s(i,j) = (c(i,j) + 1);
      end
  k = k + 1;
  end
end
imgWTxt = 'mandi.tif';
```

```
imwrite(s,imgWTxt);
```

Retrieving Code for LSB:

```
clc
s = imread(baseFileName);
height = size(s,1);
width = size(s,2);
m =  double( s(1:1:1) ) * 8  ;
k = 1;
for i = 1 : height
  for j = 1 : width
      if (k <= m)
          b(k) = mod(double(s(i,j)),2);
          k = k + 1;
      end
  end
end
binaryVector = b;
binValues = [ 128 64 32 16 8 4 2 1 ];
binaryVector = binaryVector(:);
if mod(length(binaryVector),8) ~= 0
error('Length of binary vector must be a multiple of 8.');
end
binMatrix = reshape(binaryVector,8,[]);
```

```
textString = char(binValues*binMatrix);
disp('Hidden Message is :');
disp(textString);
```

Matlab code for PVD:

```
s = c;
  height = size(c,1);
  width = size(c,2);
k = 1;
for i = 1 : height
  for j = 1 : width
      LSB = mod(double(c(i,j)), 2);
      if (k>m || LSB == b(k))
          s(i,j) = c(i,j);
      elseif(LSB == 1)
          s(i,j) = (c(i,j) - 1);
      elseif(LSB == 0)
          s(i,j) = (c(i,j) + 1);
      end
  k = k + 1;
  end
end
M=zeros(64);
k=1;
```

```
n=1;
m=1000;
j=2;
D=zeros(1,1000);
for i=1:m+1
   if i <= m
   D(k)=abs(M(j)-M(n));
   j=j+2;
   n=n+2;
   k=k+1;
   end
end
k=1;
k=1;
for i=1:m
   j=2;
   if ( 0 <D(k)<=16 )
      h=3;
   elseif ( 16 <D(k)<=32 )
      h=4;
     elseif ( 32 <D(k)<=64 )
        h=5;
           elseif ( 64 <D(k)<=128 )
              h=6;
```

```
                    elseif ( 128 <D(k)<=255 )
                        h=7;
    end
    B=zeros(1,h);
    for z=1:h
        B(z)=Sbin(z);
    end
    x=num2str(B);
    Bd=bin2dec(x);
    Dnew=Bd+2^h;
    val=Dnew-D(i);
    L=val/2;
    M(i)=M(i)-floor(L);
    M(i+1)=M(i+1)+ceil(L);

end
imgWTxt = 'moon.tif';
imwrite(s,imgWTxt);
```

Dynamic LSB Matlab code:

```
s = c;
  height = size(c,1);
  width = size(c,2);
  M=height*width;
  R=c/M;
k = 1;
for i = 1 : height
  for j = 1 : width
      LSB = mod(double(c(i,j)), 2);
      if (k>m || LSB == b(k))
          s(i,j) = c(i,j);
      elseif(LSB == 1)
          s(i,j) = (c(i,j) - 1);
      elseif(LSB == 0)
          s(i,j) = (c(i,j) + 1);
      end
  k = k + 1;
  end
  i=i+(1/R);
end
%imgWTxt = 'msgimage.png';
imgWTxt = 'mandi.tif';
imwrite(s,imgWTxt);
```

جامعة النجاح الوطنية

كلية الدراسات العليا

دراسة لمقارنة طرق إخفاء البيانات في الصور باستخدام تقنية البت الأقل أهمية وتقنية الفرق بين قيمة البكسل والبكسل المجاور

إعداد

رنا تيسير عبد الكريم الصبّاح

إشراف

د. محمد نجيب أسعد

د. لؤي ملحيس

قدّمت هذه الأطروحة استكمالاً لمتطلبات الحصول على درجة الماجستير في الرياضيات المحوسبة بكلية الدراسات العليا في جامعة النجاح الوطنية في نابلس، فلسطين.

2016

دراسة لمقارنة طرق إخفاء البيانات في الصور باستخدام تقنية البت الأقل أهمية وتقنية الفرق بين قيمة البكسل والبكسل المجاور

إعداد

رنا تيسير عبد الكريم الصبّاح

إشراف

د. محمد نجيب أسعد

د. لؤي ملحيس

الملخص

في هذه الأطروحة، نعرض أهمية إخفاء البيانات وتضمينها في الصور، ونقارن بين عدة طرق يمكن استخدامها في عملية إخفاء البيانات، منها تقنية البت الأقل أهمية وتقينة استخدام الفرق بين قيمة البكسل والبكسل المجاور له.

ومن خلال مقارنة نتائج التضمين، كانت طريقتنا التي استخدمنا فيها الفرق بين قيمة البكسل والبكسل المجاور له في إخفاء البيانات الأفضل من حيث حجم المعلومات التي يمكن إخفاؤها، والتأثير الأقل على جودة الصورة، إلى جانب السرعة في إظهار النتائج. تم عرض الطريقة وآلية عملها من خلال الأمثلة بشكل مفصّل ، وقمنا برسم خورازمية توضّح خطوات العمل، بالإضافة إلى كتابة الألغوريثم الخاص بالطريقة وتحويله إلى كود باستخدام برنامج الماتلاب وفحص النتائج ومقارنتها بالنتائج التي ظهرت معنا عند استخدام الطريقة التقليدية لإخفاء المعلومات من خلال تقينة البت الأقل أهمية، بالإضافة إلى مقارنة نتائجنا مع نتائج أبحاث سابقة استخدمت تقنيات مشابهة.