IMPLICATION OF STEGOSCRUBBING AS AN ACTIVE WARDEN TO IMAGE STEGANOGRAPHY

A PROJECT REPORT

Submitted by

HANABAL	71206205010
.VIGNESH	71206205305
.PRASANNA	71206205501

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641006

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2010

BONAFIDE CERTIFICATE

Certified that this project report entitled "IMPLICATION OF STEGOSCRUBBING AS AN ACTIVE WARDEN TO IMAGE STEGANOGRAPHY" is the bonafide work of

HANABAL	**71206205010**
VIGNESH	**71206205305**
.PRASANNA	**71206205501**

carried out the project work under my supervision.

K.R.Baskaran, M.S., M.E.,
ociate Professor,
artment of Information Technology,
naraguru College of Technology,
nnavedampatti Post,
mbatore-641006.

Dr.L.S.Jayashree,Ph.D.,
Head of department,
Department of Information Technology,
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore-641006.

mitted for viva-voice examination held on 19 / 04 / 10 .

19/4/10
ernal Examiner

19.4.10
External Examiner

ACKNOWLEDGEMENT

The exhilaration achieved on successful completion of any task should be ed with the people behind the venture. At the onset, we thank the management of college for having provided the excellent facilities to carry out the project.

We express our deep gratitude to our Principal Dr.S.Ramachandran Ph.D., for ring us in the path of triumph.

We are always thankful to our beloved Dean, Department of Computer Science Engineering, Dr.S.Thangasamy Ph.D, whose consistent support and enthusiastic lvement helped us a great deal.

We convey our sincere thanks to our project coordinator .S.Jayashree Ph.D, professor for her invaluable assistance.

We are greatly indebted to our beloved guide Mr.K.R.Baskaran M.S,M.E, ociate Professor, Department of Information Technology for her excellent lance and timely support during the course of this project.

We also feel elated in manifesting our deep sense of gratitude to all the staff and technicians in the Department of Information Technology

ABSTRACT

ABSTRACT

The terrorist groups such as Al-Qaeda are using the Internet to spread pro- rist propaganda using steganography. Steganography is the art of hiding rmation in an image file in an unsuspicious manner. Steganography is difficult to ct, often encrypted, and steganography tools free and very easy to obtain via the rnet.

Information hiding is comprised of several fascinating and complementary s; we are concerned with the important area of Steganography as it relates to ges. We propose a new method for dealing with Steganography. The aim of our ect is to write a computer program that can destroy information hidden in an ge file without degrading the image's visible quality, a process known as bbing. Our project can be implemented in the websites such as Yahoo Mail, ail etc., while uploading an image file so that covert communication gets restricted ne entry point itself.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

ABBREVIATIONS

ABBREVIATIONS

The various abbreviations used in this project report are

ASCII	American Standard Code for Information Interchange
BMP	Windows Bitmap
FBI	Federal Bureau of Investigation
GIF	Graphics Interchange Format
JPEG	Joint Photographic Experts Group
LSB	Least Significant Bit
MSB	Most Significant Bit

INTRODUCTION

1. INTRODUCTION

1.1 NEED FOR SECURITY

On July 10, 2002, USA TODAY struck fear into the hearts of terrorist-stricken Americans by publishing an article stating that terrorist groups such as al-Qaeda are using the Internet to spread pro-terrorist propaganda as well as information about attacks on the United States like the September 11th attacks. The article said these militant groups were hiding their messages in images using a method known as Steganography. Hidden within the actual bits of the file, the message is very difficult for humans and even computers to perceive. According to ***USA TODAY***, these groups are using Steganography in images on the popular auction site eBay, as well as adult websites and newsgroups.

The ***Salon.com*** article also said, however, that just because such files remain undetected does not mean that they do not exist; only that the terrorists Steganography technology may be greater than our detection technology. If that is the case, the fear that terrorists are using Steganography is quite valid and can be seen as a very real threat. Steganography is difficult to detect, often encrypted, and Steganography tools free and very easy to obtain via the Internet. So, what is to be done about the threat of terror groups using Steganography?

1.2 EXISTING SYSTEM

Steganalysis is the process of examining a message and looking for the existence of a covert message within the original message. Loosely classified into *passive* and *active* steganalysis. The methods to perform these tasks are largely statistical in nature. However it is becoming much more difficult for statistical methods to detect the presence of steganographic embedding. However, if this project's intent is to secure a network against covert communications, this project effort can be restricted to elimination of steganographic communication as a whole, rather than detection of specific instances.

1.3 PROPOSED SYSTEM

The aim of this project is to write a computer program that can destroy information hidden in an image file without degrading the image's visible quality by a process known as scrubbing.When this program scrubs the image file, it effectively destroys the hidden information, eliminating any need to decrypt it. Many corporate entities currently have safeguards in place that scan incoming email attachments with high damage potential including executable and compressed files, and the stego scrubber have similar functionality. Images will be scanned for Steganography signatures of known stego tools. If pixel side-effects from a particular algorithm can be identified, an algorithm-specific, or in formed scrubbing will be performed on the image. If the image appears to be innocent, a generic, blind scrub will be applied.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 GENERAL DESCRIPTION OF IMAGE

In the computer, an image is an array of numbers that represent light intensities at various points (*pixels*) in the image. A common image size is 640 x 480 and 256 colors (or 8 bits per pixel). Such an image could contain about 300 kilobits of data.

There are usually two type of files used when embedding data into an image. The innocent looking image, which will hold the hidden information, is a **container**. A **message** is the information to be hidden. A message may be plain-text, cipher text, other images or any thing that can be embedded in the least significant bits (LSB) of an image.

For example, suppose we have a 24-bit image 1024 x 768 (this is a common resolution for satellite images, electronic astral photographs and other high resolution graphics). This may produce a file over 2 MB in size (1024 x 768 x 24/8 = 2,359,296 bytes). All color variations are derived from three primary colors, Red, Green and Blue. Each primary color is represented by 1 byte (8 bits). 24-bit images use 3 bytes per pixel.

If information is stored in the least significant bit (LSB) of each byte, 3 bits can be a stored in each pixel. The container image will look identical to the human eye, even if viewing the picture side by side with the original. Unfortunately, 24-bit images are uncommon (with exception of the formats mentioned earlier) and quite large. They would draw attention to themselves when being transmitted across a network. Compression would be beneficial if not necessary to transmit such a file. But file compression may interfere with the storage of information.

if not necessary to transmit such a file. But file compression may interfere with the storage of information.

A pixel is an instance of color, a point in a picture.There are two kinds of compression, lossless and lossy. Both methods save storage space but may present different results when the information is uncompressed.

LOSSLESS COMPRESSION is preferred when there is a requirement that the original information remain intact (as with steganographic images). The original message can be reconstructed exactly. This type of compression is typical in GIF and BMP images.

LOSSY COMPRESSION, while also saving space, may not maintain the integrity of the original image. This method is typical in JPEG images and yields very good compression.

2.2 STEGANOGRAPHY

In this age of computers and the Internet, steganography has moved into the digital realm. An earlier form of digital steganography, known as random dot stereograms, use a computer program to hide an image by generating random dots. When correctly focused upon, the dots will reveal the hidden image. As such, stereograms do not fit into the true definition of steganography: a hidden message that is completely concealed from perception.

The form of steganography that we are using in our project is digital, meaning that its use is limited to computers. In digital steganography, the hidden message is frequently referred to as a "package". The "package" is hidden in a medium known as a "cover" or "envelope". A cover is normally a file such as a GIF, MP3, WAV, or JPEG. The most common and easily available steganography programs are for image files (e.g. GIFs). An attempt to detect or decode steganography is known as an "attack". "Scrubbing" means to simply destroy the package. "Steganalysis" is the practice of attacking steganography.***Fig2.1*** shows an example of this digital steganography. The message “If only it was this easy to detect!!!” was inserted into the image using a steganography program. No difference is visible between the cover before the information was inserted, and after.

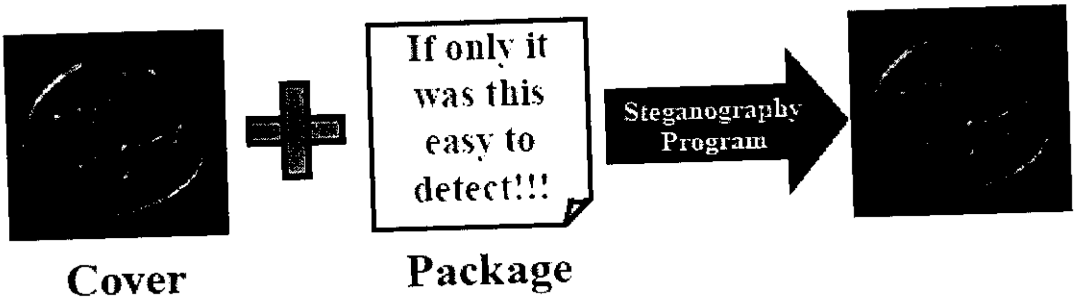


Fig 2.1: ***Illustration of steganography***

The best images for hiding information are busy images, or images with many different contrasting colors. The more contrast between different pixels, the less likely that the steganography can be seen with the human eye. For example, if an image with a single, solid color has information inserted, it will be very easy to see the difference between the original cover and the final cover. On the other hand, if an image has many different pixels with contrasting colors, the steganography will be impossible to see.

In image files, the information is hidden in the least significant bits (LSBs) of the image. These parts affect and degrade the image very little, decreasing the chance that the change can be detected.

On the Internet, the two most common image types are GIF and JPEG. JPEGs use lossy compression, which means that the data for the exact image is not stored but the computer develops an equation for its compression that approximates the original image and color values, but is not exact. This form of compression results in changes, or even losses, in the least significant bits of the pixels. Steganography programs hide information in these equations. Because of this form of estimating compression, steganography can easily be destroyed by simply opening and saving the image. The other image type, GIFs, contains and stores the exact color-values of the image. The GIF compression algorithm is non-lossy, so it does not estimate the values when it compresses the data. Because of these exact values, it is more difficult to destroy hidden data. In this image type, the data can be stored in either the least significant bits of the pixels Programs that hide information in the least significant bits of the pixels are called LSB-hiding programs.

2.3 STEGO SCRUBBING

Stego scrubbing is the technology that attempts to defeat steganography by detecting the hidden information and destroying it. Steganography tools can create stego images in which change or distortion in the cover image is not obvious to the human eye. The process of detecting a stego image is called steganalysis and the process of destroying the hidden information is called stego scrubbing.

The ultimate intent of steganography is to maximize the communications bandwidth, minimize the perceptibility of the communication and ensure robustness of the embedding. These three forces act opposition to one another.

We can conclude from the above that:

1. Imperceptible communication is not robust.
2. Embedding large messages is not robust.

Steganalysis is the process of examining an image and looking for the existence of a covert message within the original image. Our intent is to secure a network against covert communications. Our efforts can be restricted to elimination of steganographic communication as a whole.

The Elimination strategy

1. Removes the network as a channel for covert communication.
2. Allows further analysis of items in an offline situation and allows targeted analysis.
3. Forces the two communicating parties into further communication to determine what happened to the covert message.

These effects provide the following benefits for the organizations security:

1. Important information will not escape the organization forcing the covert operatives to use another more traditional and easier to detect form of communication.
2. A more detailed analysis of the original media is possible, allowing for the potential detection of covert communication and subsequent target for further investigation.
3. An aura of confusion and mistrust may be created between the sending and receiving operatives. This may force further communication between the two parties using methods that are easier to detect.

Since we are looking to remove covert communication and leave typical communicators unaware of our interference, it is also important that we also do not remove watermarking information that is used to authenticate images and protect against copyright violators.

In general, the information hiding process consists of the following steps:

1. Identification of redundant bits in a cover medium. Redundant bits are those bits that can be modified without degrading the quality of the cover medium.
2. Selection of a subset of the redundant bits to be replaced with data from a secret message.

The stego medium is created by replacing the selected redundant bits with message bits.

The modification of redundant bits can change the statistical properties of the cover medium. As a result, statistical analysis may reveal the hidden content.

There are various methods of steganalysis depending on what information is available:

Stego-only attack: Only the stego-object is available for analysis.

Known cover attack: The stego-object as well as the original medium is available. The stego-object is compared with the original cover object to detect any hidden information.

Known message attack: The hidden message and the corresponding stego-image are known. The analysis of patterns that correspond to the hidden information could help decipher such messages in future.

Known stego attack: The steganography algorithm is known and both the original and stego-object are available.

Chosen stego attack: The steganography algorithm and stego-object are known.

Chosen message attack: The steganalyst generates a stego-object from some steganography tool or algorithm of a chosen message. The goal in this attack is to determine patterns in the stego-object that may point to the use of specific steganography tools or algorithms.

The process of destroying the hidden information without degrading the image's visible quality is called scrubbing. When our program scrubs the GIF file, it effectively destroys the hidden information, eliminating any need to decrypt it. Additionally, this project is capable of distinguishing between signatures left by different steganography programs.

2.4 GENERAL DESCRIPTION OF MATLAB

Introduction

- MATLAB® is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. Using the MATLAB product, we can solve technical computing problems faster than with traditional programming languages, such as C, C++, and FORTRAN.
- MATLAB can be used in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. Add-on toolboxes (collections of special-purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems in these application areas.
- MATLAB provides a number of features for documenting and sharing your work.

Key Features

- High-level language for technical computing.
- Development environment for managing code, files, and data.
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration.
- 2-D and 3-D graphics functions for visualizing data.
- Tools for building custom graphical user interfaces.

MATLAB® provides a high-level language and development tools that let users to quickly develop and analyze algorithms and applications.

The MATLAB® Language

- The MATLAB® language supports the vector and matrix operations that are fundamental to engineering and scientific problems. It enables fast development and execution.
- With the MATLAB language, users can program and develop algorithms faster than with traditional languages and no need to perform low-level administrative tasks, such as declaring variables, specifying data types, and allocating memory. In many cases, MATLAB eliminates the need for 'for' loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code.
- At the same time, MATLAB provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, object-oriented programming (OOP), and debugging features.
- MATLAB lets users to execute commands or groups of commands one at a time, without compiling and linking, enabling to quickly iterate to the optimal solution.
- For fast execution of heavy matrix and vector computations, MATLAB uses processor-optimized libraries. For general-purpose scalar computations, MATLAB generates machine-code instructions using its JIT (Just-In-Time) compilation technology.
  This technology, which is available on most platforms, provides execution speeds that rival those of traditional programming languages.

DEVELOPMENT TOOLS

- MATLAB includes development tools that help users to implement an algorithm efficiently.

  MATLAB Editor - Provides standard editing and debugging features.

  M-Lint Code Checker - Analyzes the code and recommends changes to improve its performance and maintenance.

  MATLAB Profiler - Records the time spent executing each line of code.

  Directory Reports - Scan all the files in a directory and report on code efficiency, file differences, file dependencies, and code coverage

MATLAB provides the following types of functions for performing mathematical operations and analyzing data:

- Matrix manipulation and linear algebra
- Polynomials and interpolation
- Fourier analysis and filtering
- Data analysis and statistics
- Optimization and numerical integration
- Ordinary differential equations (ODEs)
- Partial differential equations (PDEs)
- Sparse matrix operations

- MATLAB can perform arithmetic on a wide range of data types, including doubles, singles, and integers.
- MATLAB® provides a number of features for documenting and sharing work.

- Users can integrate their MATLAB code with other languages and applications and deploy your MATLAB algorithms and applications as stand-alone programs or software modules.
- The Image Processing Toolbox provides a comprehensive set of reference-standard algorithms and graphical tools for image processing, analysis, visualization, and algorithm development.

The Advantages of MATLAB

MATLAB is a widely used tool in the electrical engineering community. It can be used for simple mathematical manipulations with matrices, for understanding and teaching basic mathematical and engineering concepts, and even for studying and simulating actual power systems and electric systems in general. The original concept of a small and handy tool has evolved to become an engineering workhorse. It is now accepted that MATLAB and its numerous Toolboxes can replace and/or enhance the usage of traditional simulation tools for advanced engineering applications.

SYSTEM DESCRIPTION

3. SYSTEM DESCRIPTION

The stego scrubbing algorithm applies differently for the Steganography system and the watermarking system. Fig 3.1depicts the overall structure of the system.

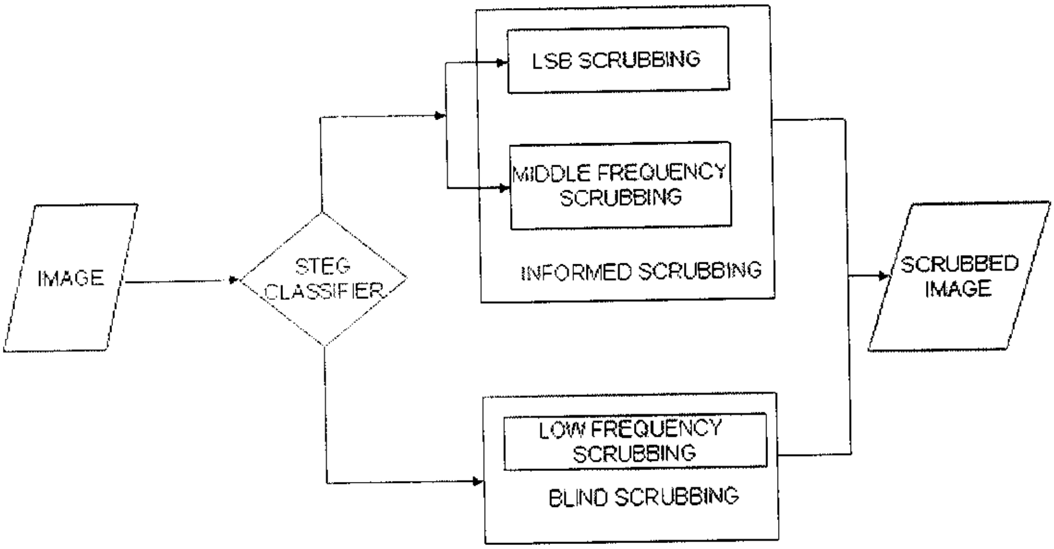


Fig 3.1: ***overall structure for stegoscrubbing***

The block diagram consists of three parts

- Steg classifier
- Informed scrubbing
- Blind scrubbing

3.1 IMPLEMENTATION OF STEGO CLASSIFIER:

Images will be scanned for steganographic signatures of known stego tools. If pixel side-effects from a particular algorithm can be identified, an algorithm -specific, or informed scrubbing will be performed on the image. If the image appears to be innocent, a generic, blind scrub will be applied.

3.2 IMPLEMENTATION OF INFORMED SCRUBBING:

If the location of steganography information is known, the locations where scrubbing has to occur can be limited, thus the amount of distortion can be limited. With informed scrubbing, there would be a requirement to have a working detector in order to guide the scrubbing process. This is possible in some cases, but small payloads may be undetectable and in these cases the process would have to resort to a generic, strong method of scrubbing.

3.3 IMPLEMENTATION OF BLIND SCRUBBING:

Blind scrubbing takes place in the absence of any information about the cover image or the embedding technique utilized. Without information about where the data could be hidden, the scrubbing probably has to be stronger or more disruptive to the image quality.

3.4 QUALITY ASSESSMENT:

Thus the performance of the stego scrubber is calculated with help of input image, stego image and scrubbed image.

ALGORITHMS IN STEGOSCRUBBING

4. ALGORITHMS IN STEGOSCRUBBING

4.1 SIGNATURES

The aspect that makes the LSB-scrubbing part of our program possible is what we call a signature. When a Steganography program inserts information, it will use the same method for every image it processes. Examples of this insertion include changing pixels in a certain pattern that is unique to a single program, such as every tenth pixel, or by distorting colors in a certain way, (for instance, changing green values in a way that no other program utilizes). In addition, a program may also insert header information, which contains information about the program used, into the image before the actual package is added. Using these patterns of pixel changing, and these headers, we can define a signature.

4.2 ALGORITHM FOR LSB HIDING

STEP 1: Start the process.

STEP 2: Load the cover image in which the message has to be hidden.

STEP3: Discrete cosine transform is applied to make it more secure.

STEP 4: Load the message that has to be hidden.

STEP 5: Convert the message into binary.

STEP 6: Get the first set of message bits that is to be stored.

STEP 7: Choose the appropriate pixels in the cover image.

STEP 8: Store the message bits in the appropriate pixels in the image.

STEP 9: If the entire message is not stored then get the next set of bits and go to STEP 5.

STEP 10: Save the resultant stego image.

STEP 11: End the process.

4.3 FLOWCHART FOR LSB HIDING

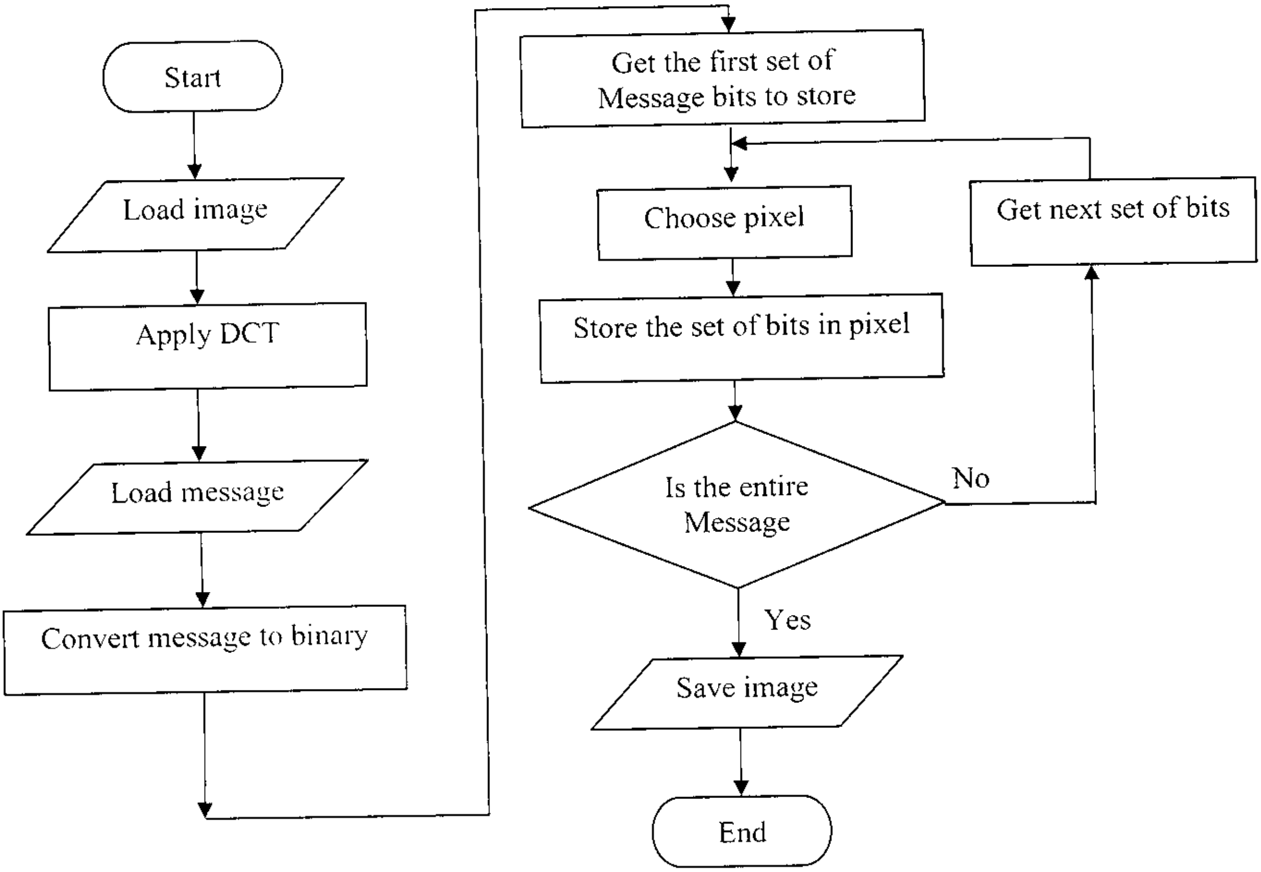


Fig 4.3 FLOWCHART FOR LSB HIDING

4.4 LSB-SCRUBBING TECHNIQUE

The LSB-scrubbing part of this program also uses opens and analyzes images. The first step is to load all of the defined signatures. It then searches the image for any color distortions, possible header information, and patterns. When it finishes this task, the program will compare possible distortions with each signature. If the number of similarities is smaller than the size of the header, the program will move on to the next signature. If the number of similarities is smaller than the header size, the program takes this result as not finding any matches, and will continue processing. By previously identifying what program is used to insert the message, our program now knows which pixels have been changed. Using this information, it locates the infected pixels. For each infected pixel, the program averages the values of the four surrounding pixels or randomizes the pixels. This average value replaces those of the infected pixel, removing the bits that held the hidden message. Because this program changed the same pixels as the Steganography program did, the recipient can locate the message by comparison. By randomizing with some uninfected pixels, the program makes it impossible for the message to be retrieved by comparing the image to the original cover. Once this step is completed, the program saves this new image, and goes on to the next one..

4.5 ALGORITHM FOR LSB SCRUBBING

STEP 1: Start the process.

STEP 2: Load the stego image in which the message has to be hidden.

STEP 3: Classify image based on signatures.

STEP 4: Check for any similarities between image and signature.

STEP 5: If LSB method identified go to STEP 6 else go to STEP 8.

STEP 6: Apply inverse DCT to an image.

STEP 7: Scrub the image by randomizing the LSB bits of pixels.

STEP 8: Save the processed image.

STEP 10: End the process.

4.6 FLOWCHART FOR LSB SCRUBBING

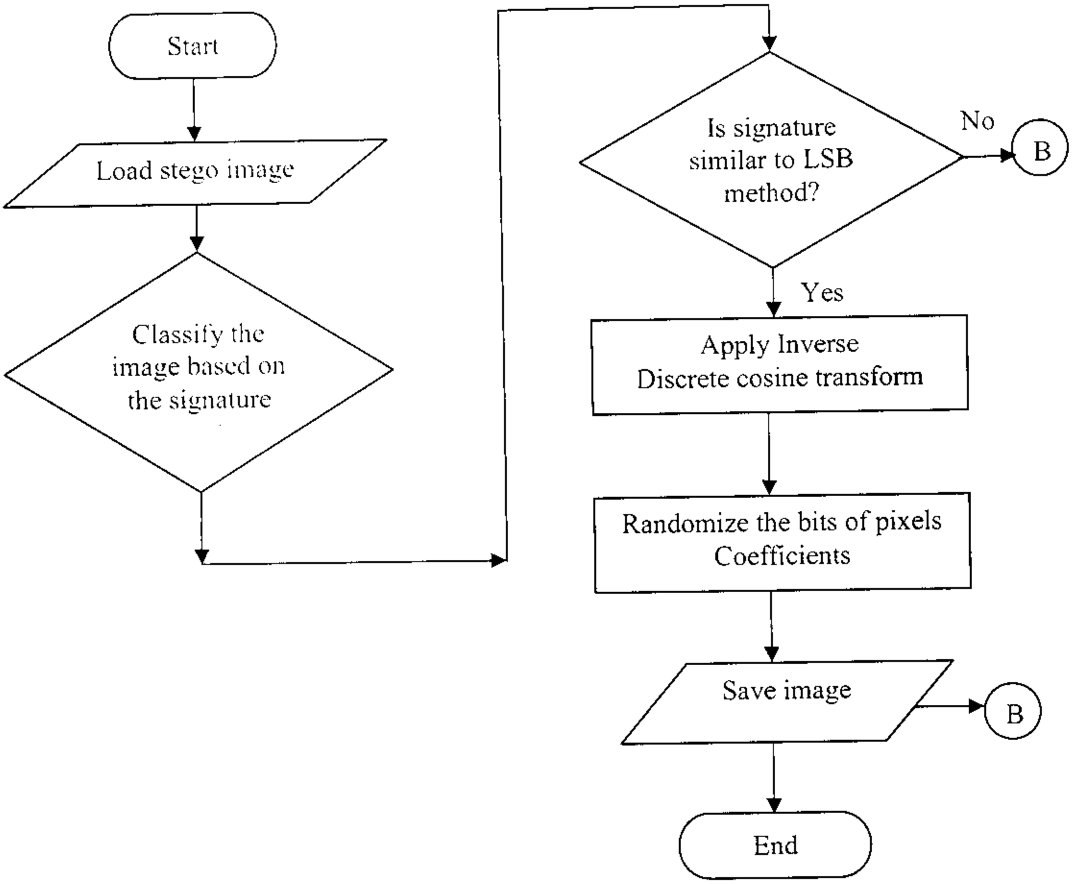

Start
Load stego image
Classify the image based on the signature
Is signature similar to LSB method?
No
B
Yes
Apply Inverse Discrete cosine transform
Randomize the bits of pixels Coefficients
Save image
B
End

Fig4.6 FLOWCHART FOR LSB SCRUBBING

4.7 DISCRETE COSINE TRANSFORM:

The DCT block computes the unitary discrete cosine transform (DCT) of each channel in the M-by-N input matrix, u.

y = dct(u)

The forward equation, for image A, is

$$b(u,v)=\frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}a(x,y)\cos\left(\frac{\pi u(2x+1)}{2N}\right)\cos\left(\frac{\pi v(2y+1)}{2N}\right)$$

The inverse equation, for image B, is

$$a(x,y)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)b(u,v)\cos\left(\frac{\pi u(2x+1)}{2N}\right)\cos\left(\frac{\pi v(2y+1)}{2N}\right)$$

The DCT has the property that, for a typical image, most of the visually significant information about the image is concentrated in just a few coefficients of the DCT. For this reason, the DCT is often used in image compression applications. For example, the DCT is at the heart of the international standard lossy image compression algorithm known as JPEG.

In the JPEG image compression algorithm, the input image is divided into 8-by-8 or 16-by-16 blocks, and the two-dimensional DCT is computed for each block. The DCT coefficients are then quantized, coded, and transmitted. The JPEG receiver (or JPEG file reader) decodes the quantized DCT coefficients, computes the inverse two-dimensional DCT of each block, and then puts the blocks back together into a single image. For typical images, many of the DCT coefficients have values close to zero and these coefficients are known as middle frequency coefficients. Thus the middle frequency DCT coefficients are replaced with message. This is known as DCT middle frequency hiding.

4.8 ALGORITHM FOR MIDDLE FREQUENCY HIDING

STEP 1: Start the process.

STEP 2: Load the cover image in which the message has to be hidden.

STEP3: Discrete cosine transform is applied to make it more secure.

STEP 4: Load the message that has to be hidden.

STEP 5: Convert the message into binary.

STEP 6: Get the first set of message bits that is to be stored.

STEP 7: Choose the middle frequency pixels coefficients in the discrete Cosine transformed cover image.

STEP 8: Replace the message bits with the middle frequency pixels in the DCT original image.

STEP 9: If the entire message is not stored then get the next set of bits and go to STEP 5.

STEP 10: Save the resultant stego image.

STEP 11: End the process.

4.9 FLOWCHART FOR MIDDLE FREQUENCY HIDING

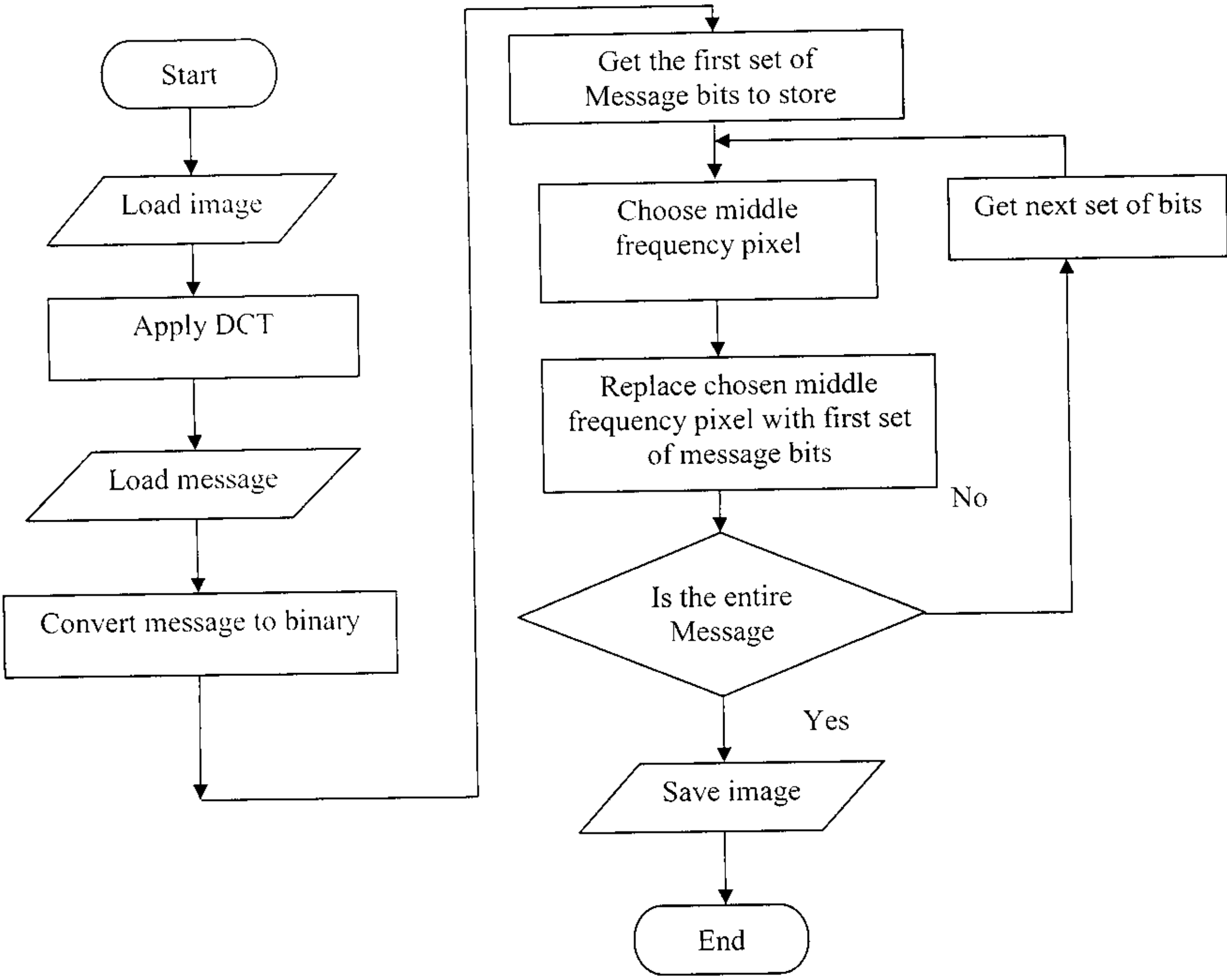

Start
Load image
Apply DCT
Load message
Convert message to binary
Get the first set of Message bits to store
Choose middle frequency pixel
Get next set of bits
Replace chosen middle frequency pixel with first set of message bits
No
Is the entire Message
Yes
Save image
End

Fig 4.9 FLOWCHART FOR MIDDLE FREQUENCY HIDING

4.10 ALGORITHM FOR MIDDLE FREQUENCY SCRUBBING

STEP 1: Start the process.

STEP 2: Load the stego image in which the message has to be hidden.

STEP 3: Classify image based on signatures.

STEP 4: Check for any similarities between image and signature.

STEP 5: If Middle frequency hiding method identified go to STEP 6 else go to STEP 8.

STEP 6: Apply inverse DCT to an image.

STEP 7: Scrub the image by randomizing the middle frequency pixel Coefficient with other coefficients.

STEP 8: Save the processed image.

STEP 9: End the process.

4.11 FLOWCHART FOR MIDDLE FREQUENCY SCRUBBING

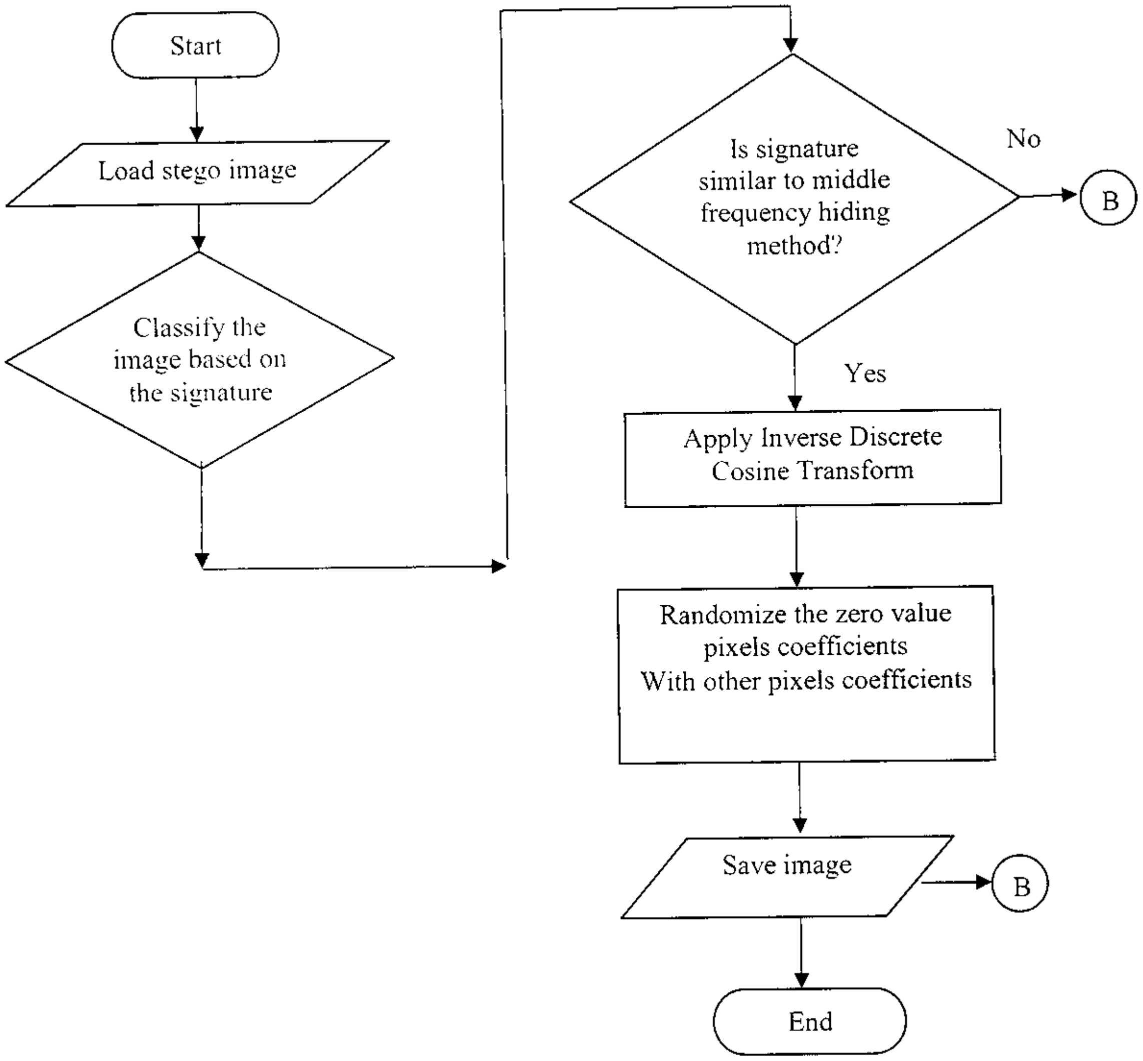

Start
Load stego image
Classify the image based on the signature
Is signature similar to middle frequency hiding method?
No
B
Yes
Apply Inverse Discrete Cosine Transform
Randomize the zero value pixels coefficients With other pixels coefficients
Save image
B
End

Fig 4.11 FLOWCHART FOR MIDDLE FREQUENCY SCRUBBING

4.12 DISCRETE WAVELET TRANSFORM

The DWT block computes the discrete wavelet transform (DWT) of each column of a frame-based input. By default, the output is a sample-based vector or matrix with the same dimensions as the input. Each column of the output is the DWT of the corresponding input column.

For the same input, the DWT block and the dwt function, in the Wavelet Toolbox, do not produce the same results. Because the block set is designed for real-time implementation and the toolbox is designed for analysis, the products handle boundary conditions and filter states differently.

To make the output of the DWT block match the output of the dwt function, complete the following steps:

- For the dwt function, set the boundary condition to zero-padding by typing dwtmode('zpd') at the MATLAB command prompt.

- To match the latency of the DWT block, which is implemented using FIR filters, add zeros to the input of the dwt function. The number of zeros we add must be equal to the half the filter length.

DWT transforms the original image into many components such as low frequency image, vertical and horizontal frequency image components. Thus hiding message in low frequency component is called low frequency hiding.

SINGLE LEVEL 1-D WAVELET TRANSFORMS:

Syntax

```
[cA,cD] = dwt(X,'wname')
[cA,cD] = dwt(X,'wname','mode',MODE)
[cA,cD] = dwt(X,Lo_D,Hi_D)
[cA,cD] = dwt(X,Lo_D,Hi_D,'mode',MODE)
```

Description

The dwt command performs a single-level one-dimensional wavelet decomposition with respect to either a particular wavelet or particular wavelet decomposition filters (Lo_D and Hi_D) that you specify.

[cA,cD] = dwt(X,'wname') computes the approximation coefficients vector cA and detail coefficients vector cD, obtained by a wavelet decomposition of the vector X. The string 'wname' contains the wavelet name. [cA,cD] = dwt(X,Lo_D,Hi_D) computes the wavelet decomposition as above, given these filters as input: Lo_D is the decomposition low-pass filter. Hi_D is the decomposition high-pass filter.Lo_D and Hi_D must be the same length.

4.13 ALGORITHM FOR LOW FREQUENCY HIDING

STEP 1: Start the process.

STEP 2: Load the cover image in which the message has to be hidden.

STEP3: Discrete wavelet transform is applied to make it more secure.

STEP 4: Load the message that has to be hidden.

STEP 5: Convert the message into binary.

STEP 6: Get the first set of message bits that is to be stored.

STEP 7: Choose the low frequency pixels coefficients in the discrete
Wavelet transformed cover image.

STEP 8: Store the message bits in the low frequency pixels in the
DWT original image.

STEP 9: If the entire message is not stored then get the next set of bits and
Go to **STEP** 5.

STEP 10: Save the resultant stego image.

STEP 11: End the process.

4.14 FLOWCHART FOR LOW FREQUENCY HIDING

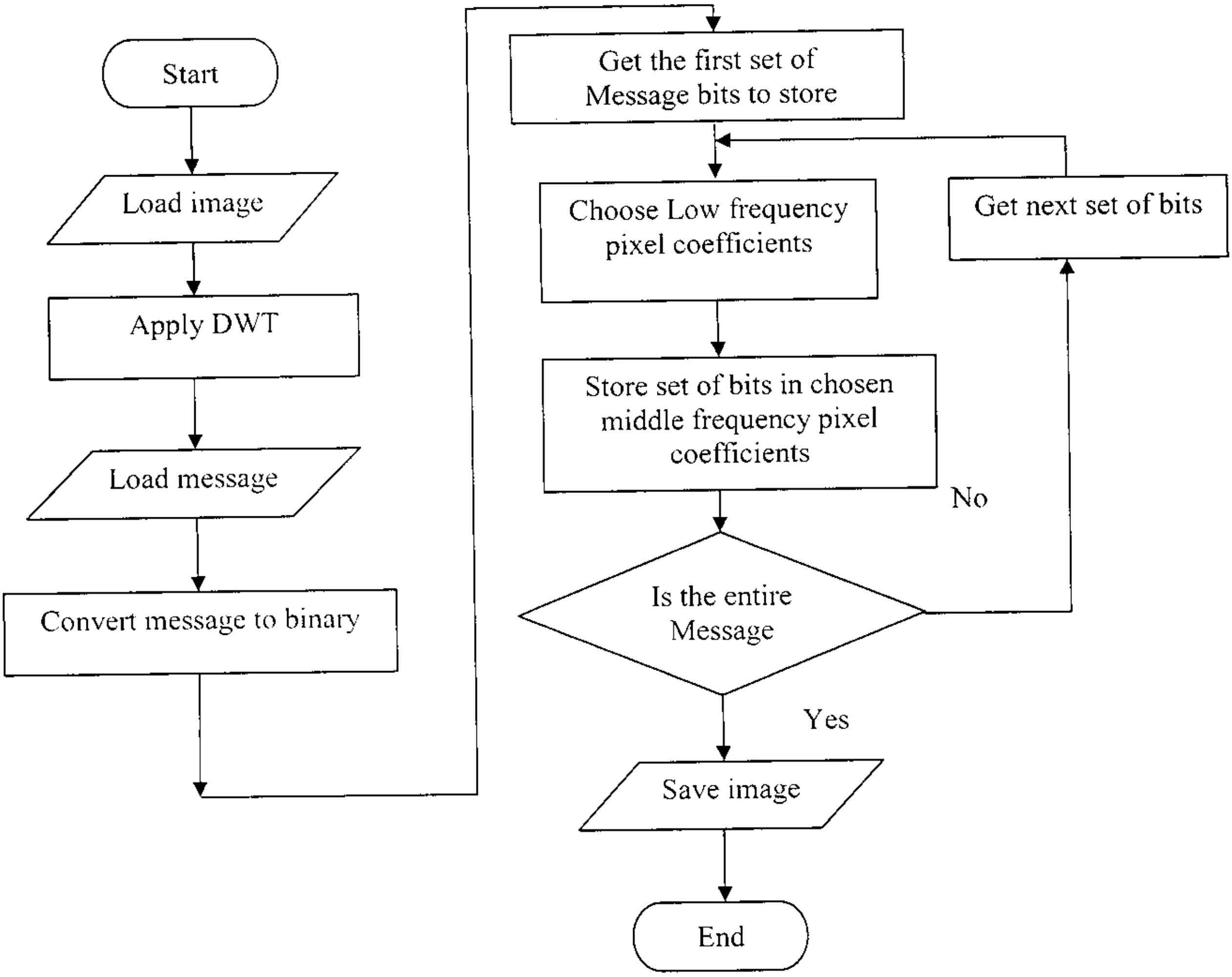

Start
Load image
Apply DWT
Load message
Convert message to binary
Get the first set of Message bits to store
Choose Low frequency pixel coefficients
Store set of bits in chosen middle frequency pixel coefficients
Is the entire Message
No
Get next set of bits
Yes
Save image
End

Fig 4.14 FLOWCHART FOR LOW FREQUENCY HIDING

4.15 ALGORITHM FOR LOW FREQUENCY SCRUBBING

STEP 1: Start the process.

STEP 2: Load the stego image in which the message has to be hidden.

STEP 3: Classify image based on signatures.

STEP 4: Check for any similarities between image and signature.

STEP 5: If Low frequency hiding method identified go to STEP 6 else
Go to STEP 8.

STEP 6: Apply inverse DWT to an image.

STEP 7: Scrub the image by randomizing the low frequency pixel
Coefficient in DWT image with others.

STEP 8: Save the processed image.

STEP 9: End the process.

4.16 FLOWCHART FOR LOW FREQUENCY SCRUBBING

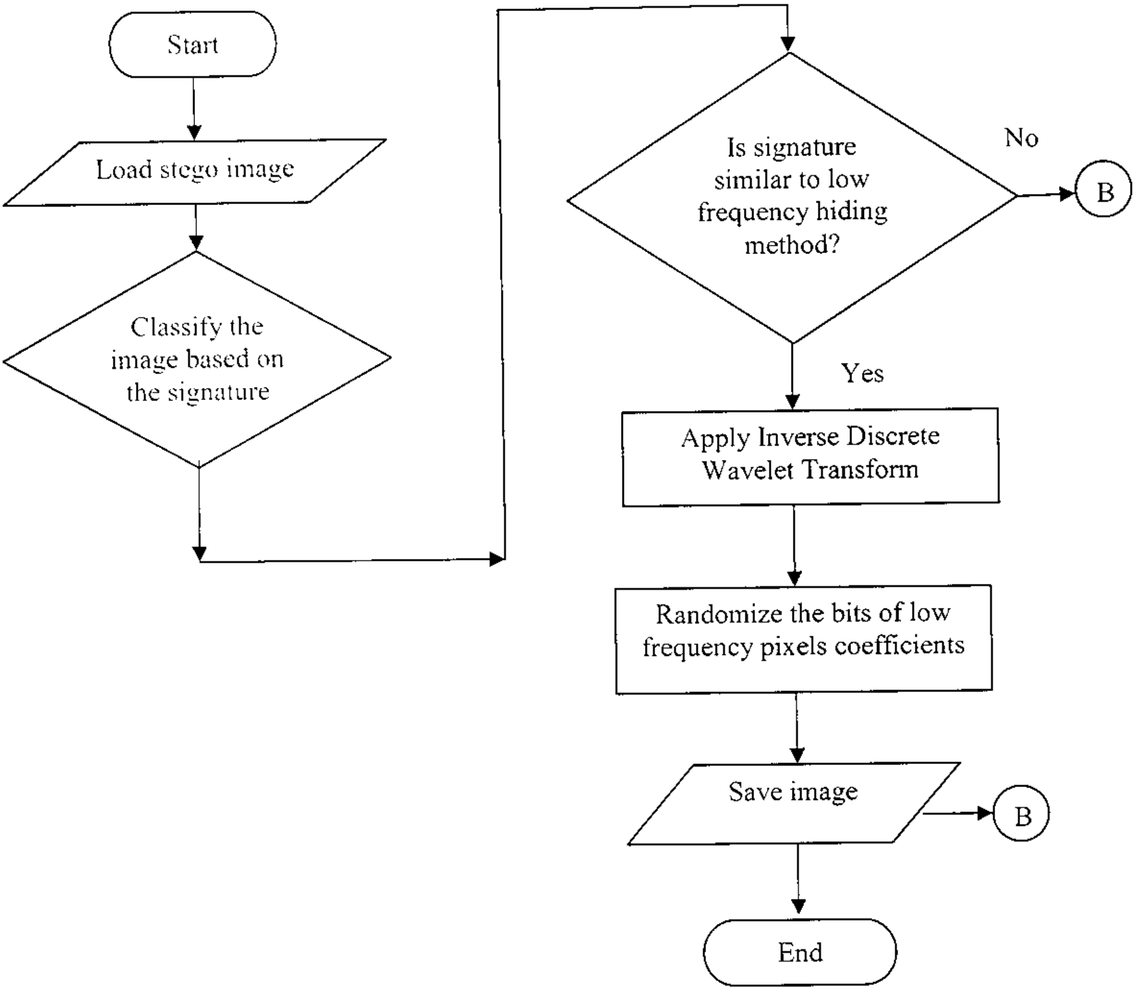

Start
Load stego image
Classify the image based on the signature
Is signature similar to low frequency hiding method?
No
B
Yes
Apply Inverse Discrete Wavelet Transform
Randomize the bits of low frequency pixels coefficients
Save image
B
End

Fig 4.16 FLOWCHART FOR LOW FREQUENCY SCRUBBING

4.17 Advantages and Limitations

Advantages

- Business people could use it to prevent employees from using Steganography to leak confidential information.
- Our program can be implemented in large commercial Internet-based companies, such as eBay which has the threat of terrorists using images on the site to pass information.
- The robustness, quality, intensity & brightness are unaltered even after encoding, decoding and scrubbing.

Limitations

- Carrier image should be at least double the size of the message file. For example, if the message file size is 450KB then the carrier image size should be at least 900KB. This is because that 1 byte of data is stored in 2 color values.
- Steganography technique implemented other than images cannot be scrubbed.

4.18 System Requirements

The minimum system requirements for the code to execute effectively Are as follows:

- **HARDWARE REQUIREMENTS:**
    - Pentium IV
    - 512 MB RAM
    - 2 GB of free space.
- **SOFTWARE REQUIREMENTS:**
    - MATLAB R2006B
    - Windows XP

CONCLUSION

5. CONCLUSION

Our stegoscrubbing implementation wins over the challenging terrorist communication technique the steganography. The fact that we can destroy information hidden in an image file without degrading the image's visible quality by a process known as scrubbing was successfully carried out and the quality assessment results also prove the same. This indicates that stegoscrubbing can be used as an active warden to image steganography.

FUTURE SCOPE

6. FUTURE SCOPE

Our project has wide future scope in the internet. Our stego scrubbing algorithm could be implemented in the Attach file option of the most common mail service enabled websites so that covert mails would be restricted at the entry point itself.

APPENDIX I

APPENDIX I

A1 CODING

FORMCREATION – STEGOSCRUBBING

MODULE FINAL.M

```
clc
close all
clear all
warning off
choice=0;
possibility=6;


while choice~=possibility,
        choice=menu('STEGO SCRUBBBING','INPUT','EMBEDDING-
CLASIFICATION','EXTRACTING','SCRUBBING','PERFORMANCE','EX
IT');
                if choice==1

[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.jpeg;*.fig;*.gif;
*.png;*.jpg;*.pgm','IMAGE Files (*.bmp,*.tif,*.jpeg,*.gif, *.png,*.jpg;
*.pgm)'},'Choose an Image');
            [I,map]=imread(strcat(pathname,namefile));
            I=I(:,:,1);
            I=imresize(I,[500 500]);
            txt=input('Enter the text::');
            [row col]=size(I);
                 end

                if choice==2
           choi=0;
           poss=4;

                while choi~=poss,
                      choi=menu('Embedding','DCT-LSB','DCT-Middle
 Freq','DWT','Back');
```

```
  if choi==1

           [img] = embedd(I,txt);
           [row col]=size(img);
  end

  if choi==2

           [img,Z]=embe_dct(I,txt);
           [row col]=size(img);
  end

  if choi==3

           [img] = embedd_dwt(I,txt);
           ttxt=length(txt);
           [row col]=size(img);
           img(row)=ttxt;
           ssiz=img(row);
  end
          if choi==4

  end
        imwrite(uint8(img),'Embedded Image.jpg','jpg')
end
    imag=round(dct2(img));
    siz=img(row);

for i=1:row
      j=i+1;
         if i==row
              j=1;
         end
           y1(i)=(img(i)+img(j))/2;
      y(i)=img(i);
end
     [r c]=size(y1);
for i=1:c
    T(i)=(y(i)-y1(i))/y1(i);
end
```

```
if choice==3
                  h=waitbar(0,'Extracting the  Data');

                 for i=1:10000
                     waitbar(i/10000)
                 end
                  close(h)
    if m==1
           bsiz = 8*siz;
           n = numel(imag);

           I1 = reshape(imag,1,n);
           I2 = round(abs(I1(1:bsiz)));
           p = 2;
           h = 2^0;
            rb = zeros(1,bsiz);
                   for k = 1:bsiz
                                 I2(k) = round(I2(k));
                                 r = rem(I2(k),p);
                              if r >= h
                                   rb(k) = 1;
                              end
                   end
          rbi = (dec2bin(rb,1))';
          rbin = reshape(rbi,siz,8);
          rectxt = (bin2dec(rbin))';

          steg=char(rectxt);
          msgbox(steg)

     elseif m==2

           [rectxt] = recoverbp(imag,siz,Z);
           steg=char(rectxt);
             msgbox(steg)

     elseif m==3

         recover_dwt(I,img,ssiz)
```

```
  end
    end

if choice==4
  [imag]=randamize(img);

  imwrite(uint8(imag),'Scrubbed image.jpg','jpg')
  I=imag;
 end

if choice==5
  [row col]=size(I);
  steg=double(imread('Embedded Image.jpg'));
  scru=double(imread('Scrubbed image.jpg'));
  for i=1:row
    for j=1:col
        x(i,j)=double(I(i,j));
        y(i,j)=steg(i,j);
        x_c(i,j)=scru(i,j);
        num(:,i)=(x(i,j)-y(i,j))^2;
        den(:,i)=(x(i,j)-x_c(i,j))^2;
     end
  end
  disp('SNR Value is::')
   snr=10*(log(sum(num)/sum(den)));
   disp(snr)
  if choice==6
    close all
  end
    end
 end
```

MODULE CLASSIFICATION.M

```
function [m]=classification(T,y1,row,siz)

    P = cdf('t',T,y1);
    pr=(1-P)*100;
        c=8*siz;
    I1=zeros(row,row);
    I=[30 28 22 15 0];
        h=waitbar(0,'Classifying the Image');
        for i=1:10000
       waitbar(i/10000)
    end
    close(h)
    if row==450
          p=rand(c);
          pro=p(1:c);
          pro1=pro+99;
          pro = [I(1,:) pro1 I1(c+1:100)];
          plot(pro)
          grid on
        xlabel('Analysed position in image in percent')
        ylabel('Probability of embedding in percent')
           axis([0 100 0 110])
          m=1;
          elseif row==512
           if siz==0;
              return;
           end
              for plo=1:c
              pro(plo)=unifrnd(20,80);
           end
          plot(pro)
          xlabel('Analysed position in image in percent')
        ylabel('Probability of embedding in percent')
      else
        m=3;
    end
```

MODULE

INFORMED SCRUBBING.M

EMBEDDING:DCT_LSB

```
function [img] = embedd(I,txt)
b=1;
I=imresize(I,[450 450]);

I=round(dct2(I));

si_tx=length(txt);
I1=I;
N = 8*numel(txt);          % number of elements in a array(text)
S = numel(I);              % number of elements in a image
if N > S
   warning('Text truncated to be within size of image')
end
p = 2^b;
dim = size(I);
I2 = round(abs(I1(1:N)));       %gettimg I2 depending upon the N..
si = sign(I1(1:N));
  for k = 1:N
     if si(k) == 0
        si(k) = 1;
     end
     I2(k) = round(I2(k));
      if mod((I2(k)),p) >= b
         I2(k) = I2(k) + b;
      end
  end
 bt = dec2bin(txt,8);
 bint = reshape(bt,1,N); %bint is in char
 d = b*48;              %ascii
 bi = (b*bint) - d;     %char to double
 I3 = double(I2) - bi;
 I5 = [I3 I1(N+1:S)];
 intl = reshape(I5,dim);
```

```
img=idct2(intl);
 [row col]=size(img);
img(row)=si_tx;
```

RECOVER: DCT_LSB

```
[row col]=size(img);
img(row)=si_tx;

    bsiz = 8*siz;
                n = numel(imag);
                 I1 = reshape(imag,1,n);
                I2 = round(abs(I1(1:bsiz)));
                p = 2;
                h = 2^0;
                 rb = zeros(1,bsiz);
                      for k = 1:bsiz
                                I2(k) = round(I2(k));
                                r = rem(I2(k),p);
                              if r >= h
                                 rb(k) = 1;
                              end
                      end
               rbi = (dec2bin(rb,1))';
               rbin = reshape(rbi,siz,8);
               rectxt = (bin2dec(rbin))';

               steg=char(rectxt);
               msgbox(steg)
```

EMBEDDING: DCT_MIDDLE FREQUENCY

```
function [X2,Z]=embe_dct(I,txt)
I=imresize(I,[512 512]);
          X=I;
          si_tx=length(txt);
          Y = dct2(X);
          Y1=round(Y);
          Z=find(Y1==0);
```

```
        N=8*numel(txt);
         B=dec2bin(txt,8);
          B1=reshape(B,1,N);
          d=48;
         B2=B1-d;
        for i=1:length(B2)
           Y1(Z(i))=B2(i);
        end
          X2 = idct2(Y1);
          [row col]=size(X2);
         X2(row)=si_tx;
```

RECOVER: DCT_MIDDLE FREQUENCY

```
function [rectxt] = recoverbp(I,siz,Z)
 I1=I;
bsiz = 8*siz;
for j = 1:bsiz
   base=Z(j);
   rb(j)=I1(base);
end
rb=abs(rb);
rbi = (dec2bin(rb,1))';
 rbi=rbi(1:bsiz);
rbin = reshape(rbi,siz,8);
rectxt = (bin2dec(rbin))';
return
```

INFORMED SCRUBBING:

```
function [new_image]=randamize(I)
[row col dim]=size(I);
I=imresize(I,[100 100]);
dim=size(I);
S=numel(I);
p=randperm(S);

I1=double(reshape(I,1,S));
h=waitbar(0,'Randomizing the Pixels');

for i=1:S
```

```
   waitbar(i/S)
    Y=p(i);
    X(i)=I1(Y);
 end
close(h)
new_image=uint8(reshape(X,dim));
new_image=imresize(new_image,[row col]);
figure;
imshow(new_image)
title('Pixel Randomized Image')
```

MODULE

BLIND SCRUBBING.M

EMBEDDING: DWT_LOW FREQUENCY

```
function [idct_I]=embedd_dwt(I,txt)

[cA1,cH1,cV1,cD1] = dwt2(I,'db1');
figure;title('DWT Image');
subplot(2,2,1),imshow(uint8(cA1))
subplot(2,2,2),imshow(uint8(cH1))
subplot(2,2,3),imshow(uint8(cV1))
subplot(2,2,4),imshow(uint8(cD1))
 N = 8*numel(txt);
 tot_num=N;
S = numel(cA1);
if N > S
   warning('Text truncated to be within size of image')
end

I1 = reshape(cA1,1,S);
dim = size(cA1);
 I2 = round(abs(I1(1:tot_num)));


bit_con = dec2bin(txt,8);
```

```
d = 48;
bin_val = (bit_int) - d;
bin=bin_val;
alpha=1;

for i=1:length(bin)
   if bin(i)==1
      test_Im(i)=I2(i)+alpha;
   else
      test_Im(i)=I2(i)-alpha;
   end
end
 I5 = [test_Im I1(tot_num+1:S)];
emb_img = reshape(I5,dim);
idct_l = idwt2(emb_img,cH1,cV1,cD1,'db1');
idct_l=imresize(idct_l,[500 500]);
figure;imshow(uint8(idct_l));title('Embedded :Low frequency DWT Image')
```

RECOVER: DWT_LOW FREQUENCY

```
function recover_dwt(I1,idct_l,siz)
 [cA1,cH1,cV1,cD1] = dwt2(I1,'db1');
im_cr=cA1;
[cA2,cH2,cV2,cD2] = dwt2(idct_l,'db1');
bin=siz*8;
 S1=numel(cA2);
mar1 = reshape(cA2,1,S1);
 for x=1:bin

      if mar1(x)-im_cr(x)>=0
         mar(x)=1;
      else
         mar(x)=0;
      end

 N = 8*siz;
  m=mar(1:N);
 rec_bin = (dec2bin(m,1))';
 rec_val = reshape(rec_bin,siz,8);
```

```
rectxt1 = (bin2dec(rec_val))';
 hidden_txt=char(rectxt1);
  msgbox(hidden_txt)
```

BLIND SCRUBBING:

```
function [new_image]=randamize(I)
[row col dim]=size(I);
I=imresize(I,[100 100]);
dim=size(I);
S=numel(I);
p=randperm(S);

I1=double(reshape(I,1,S));
h=waitbar(0,'Randomizing the Pixels');

for i=1:S
   waitbar(i/S)
    Y=p(i);
    X(i)=I1(Y);

end
 close(h)

new_image=uint8(reshape(X,dim));
new_image=imresize(new_image,[row col]);
figure;
imshow(new_image)
title('Pixel Randomized Image')
```

MODULE

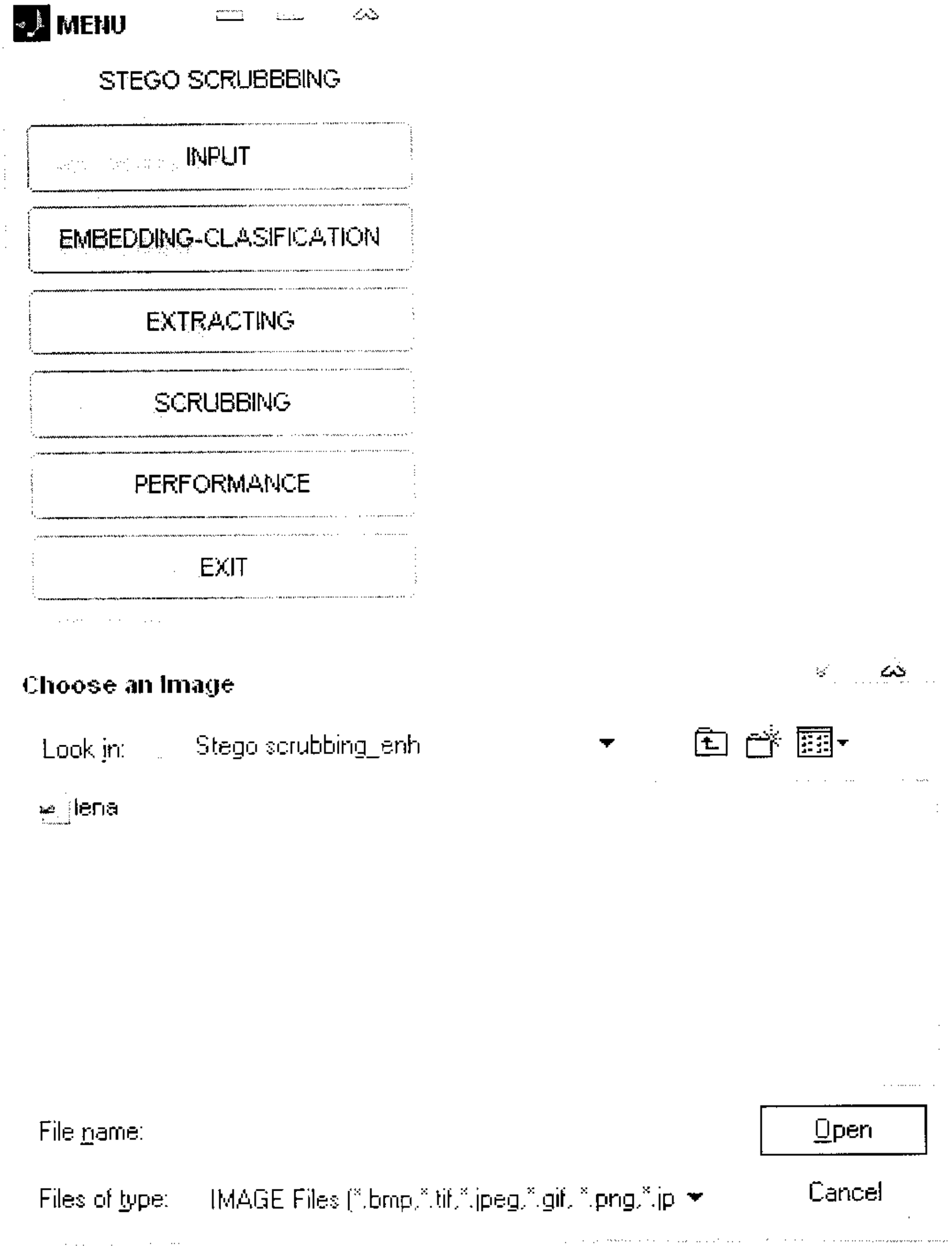
QUALITY ASSESSMENT.M

```
[row col]=size(I);
        steg=double(imread('Embedded Image.jpg'));
        scru=double(imread('Scrubbed image.jpg'));

        for i=1:row
           for j=1:col
               x(i,j)=double(I(i,j));
               y(i,j)=steg(i,j);
               x_c(i,j)=scru(i,j);
               num(:,i)=(x(i,j)-y(i,j))^2;
               den(:,i)=(x(i,j)-x_c(i,j))^2;

           end
        end
        disp('SNR Value is::')

        snr=10*(log(sum(num)/sum(den)));
         disp(snr)
```

APPENDIX II

APPENDIX II

A2 IMPLEMENTATION RESULTS

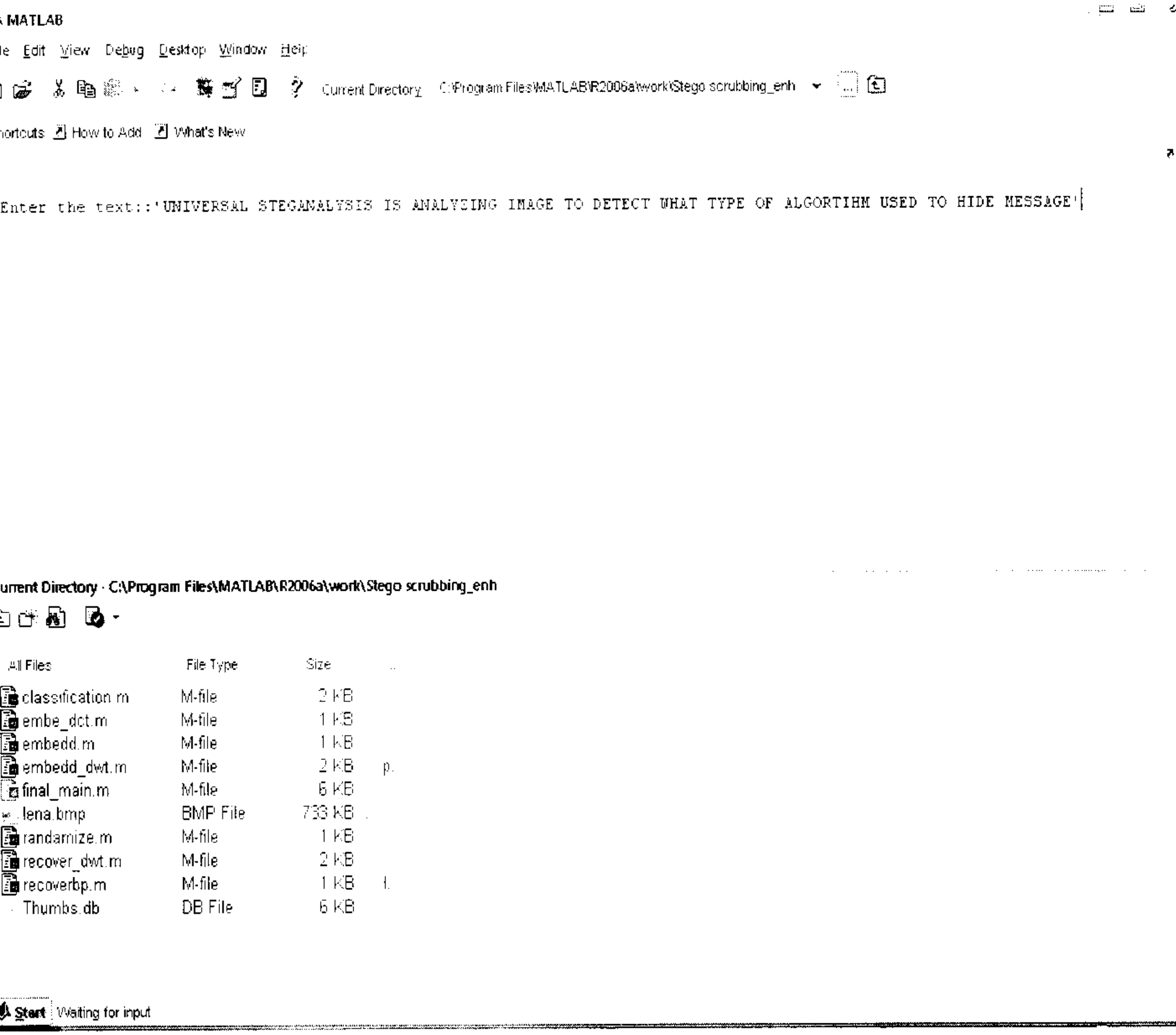


Look in: Stego scrubbing_enh

File name:

Files of type: IMAGE Files (*.bmp,*.tif,*.jpeg,*.gif, *.png,*.jp

Fig A2.1: MAIN MENU WITH INPUT IMAGE SELECTION


MATLAB
le Edit View Debug Desktop Window Help
Current Directory C:\Program Files\MATLAB\R2006a\work\Stego scrubbing_enh
nortcuts How to Add What's New

```
Enter the text::'UNIVERSAL STEGANALYSIS IS ANALYSING IMAGE TO DETECT WHAT TYPE OF ALGORTIHM USED TO HIDE MESSAGE'
```

urrent Directory - C:\Program Files\MATLAB\R2006a\work\Stego scrubbing_enh

All Files	File Type	Size
classification.m	M-file	2 KB
embe_dct.m	M-file	1 KB
embedd.m	M-file	1 KB
embedd_dwt.m	M-file	2 KB
final_main.m	M-file	6 KB
lena.bmp	BMP File	733 KB
randamize.m	M-file	1 KB
recover_dwt.m	M-file	2 KB
recoverbp.m	M-file	1 KB
Thumbs.db	DB File	6 KB

Fig A2.2: INPUT FOR ENCODING MESSAGE INTO IMAGE

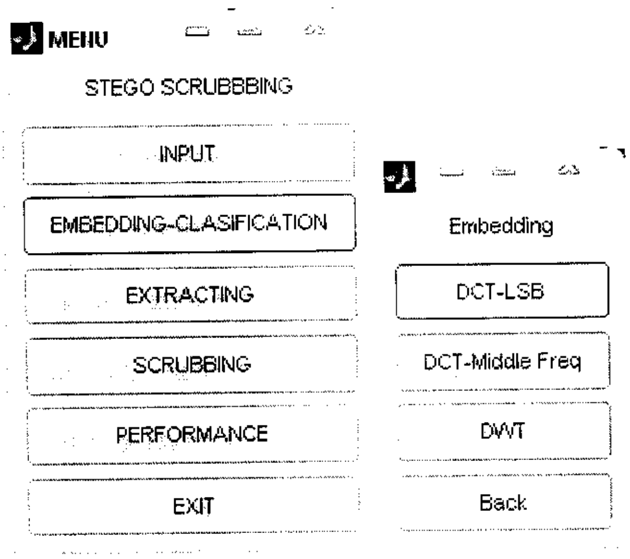


Fig A2.3: EMBEDDING TECHNIQUE SELECTION

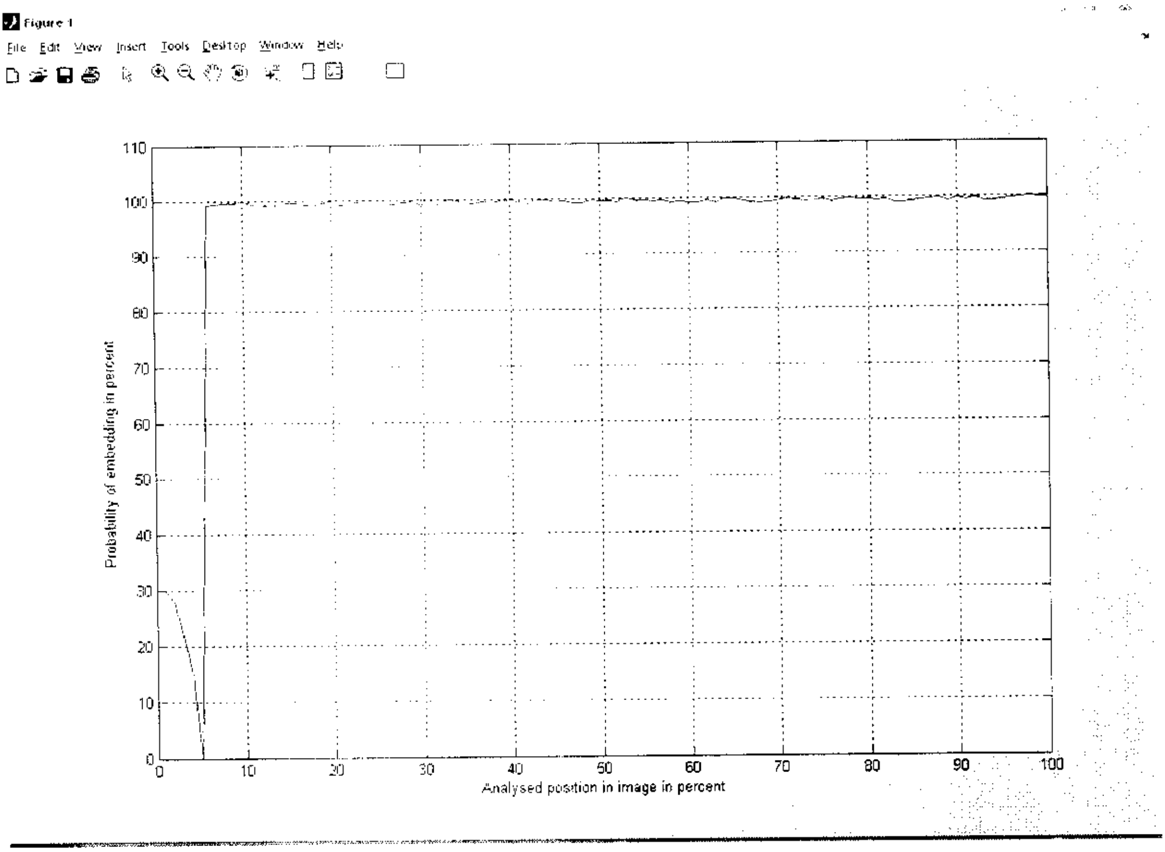


Fig A2.4: GRAPH SHOWING LSB TECHNIQUE USED

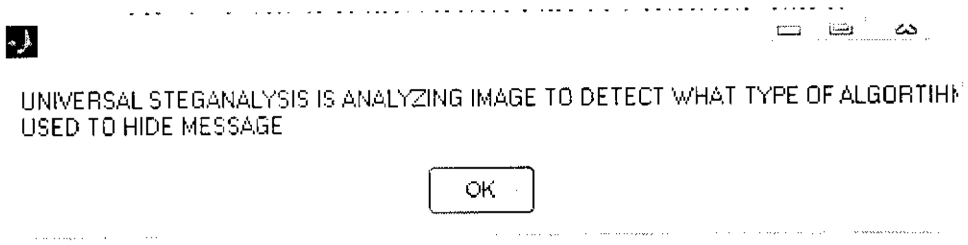

UNIVERSAL STEGANALYSIS IS ANALYZING IMAGE TO DETECT WHAT TYPE OF ALGORTIHM
USED TO HIDE MESSAGE

Fig A2.5: EXTRACTED MESSAGE BEFORE SCRUBBING

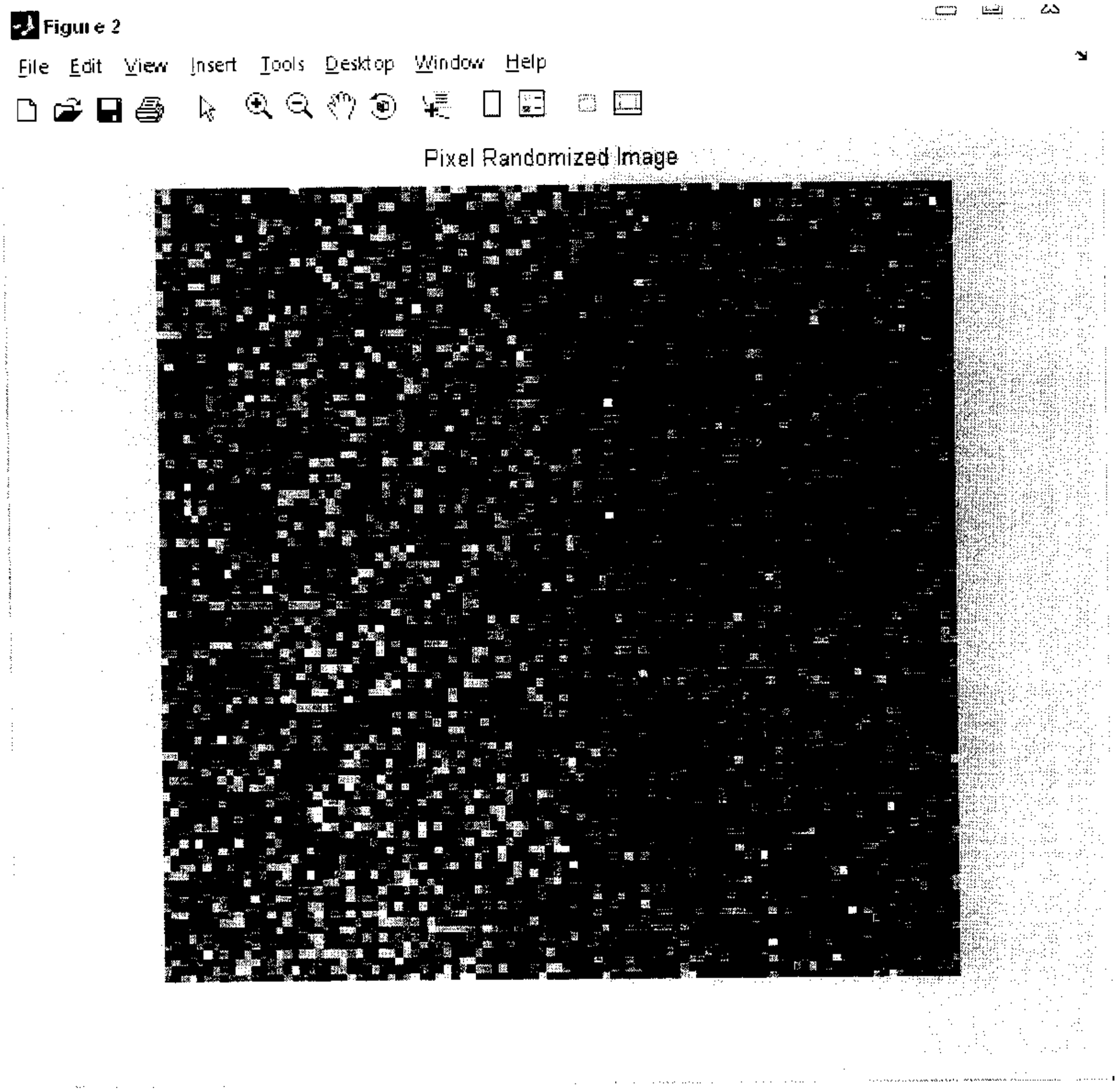
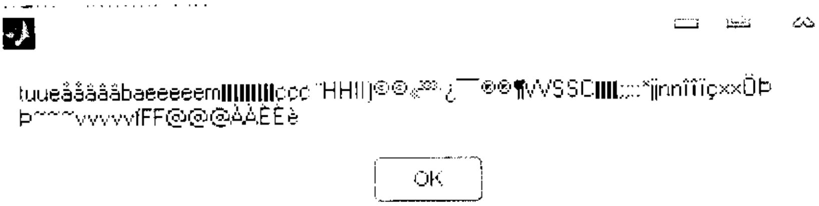
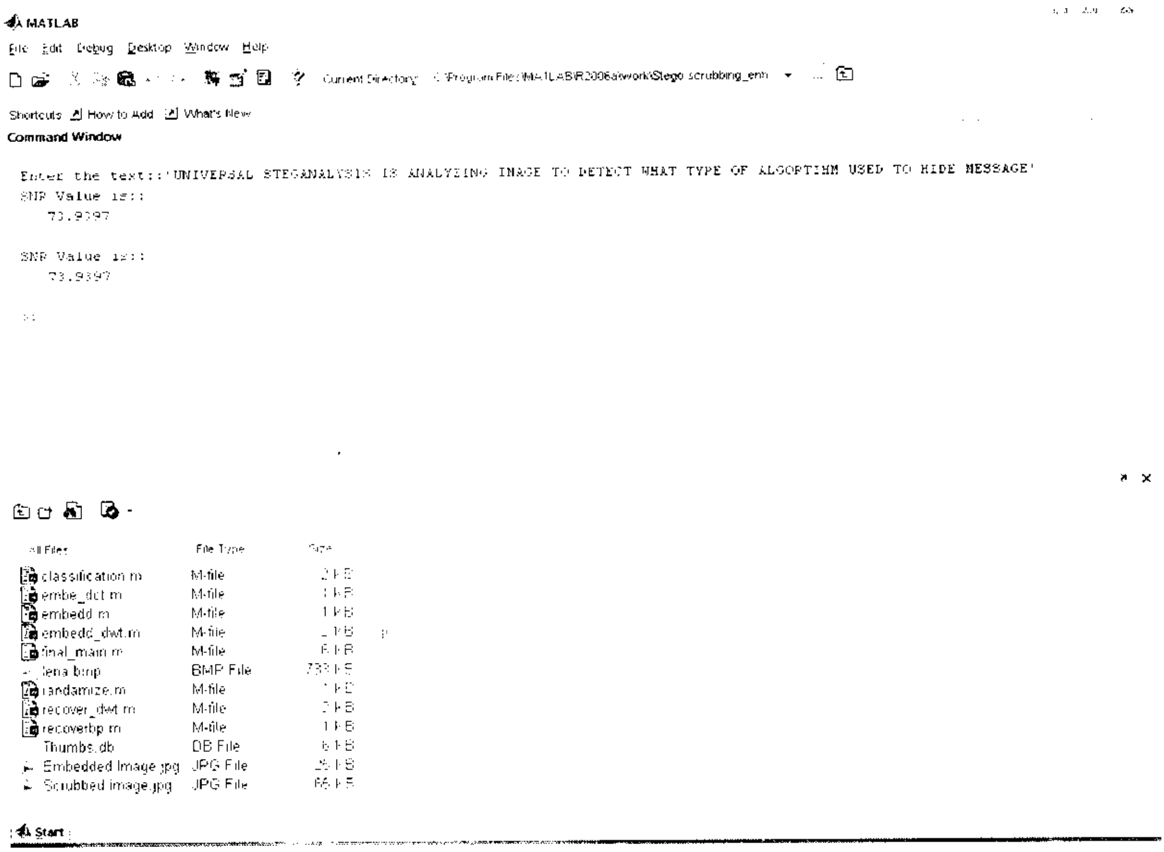


Fig A2.6: SCRUBBED DCT IMAGE



tuueåååååbaeeeeeemllllllllççc¨HHII]©©«ººº¿¯®®¶VVSSCllll;;;:^jjnnîîîçxxÖÞ
Þ~~~~vvvvvfFF@@@ÀÀÊÊè

Fig A2.7: HIDDEN MESSAGE AFTER SCRUBBING


MATLAB
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
Command Window
Start

Current Directory: C:\Program Files\MATLAB\R2006a\work\Stego scrubbing_enh

```
Enter the text::'UNIVERSAL STEGANALYSIS IS ANALYZING IMAGE TO DETECT WHAT TYPE OF ALGORTIHM USED TO HIDE MESSAGE'
SNR Value is::
   73.9397

SNR Value is::
   73.9397

>>
```

All Files	File Type	Size
classification.m	M-file	2 KB
embe_dct.m	M-file	1 KB
embedd.m	M-file	1 KB
embedd_dwt.m	M-file	1 KB
final_main.m	M-file	6 KB
lena.bmp	BMP File	733 KB
randamize.m	M-file	1 KB
recover_dwt.m	M-file	2 KB
recoverbp.m	M-file	1 KB
Thumbs.db	DB File	6 KB
Embedded Image.jpg	JPG File	25 KB
Scrubbed image.jpg	JPG File	66 KB

Fig A2.8: QUALITY ASSESSMENT-DCT_LSB

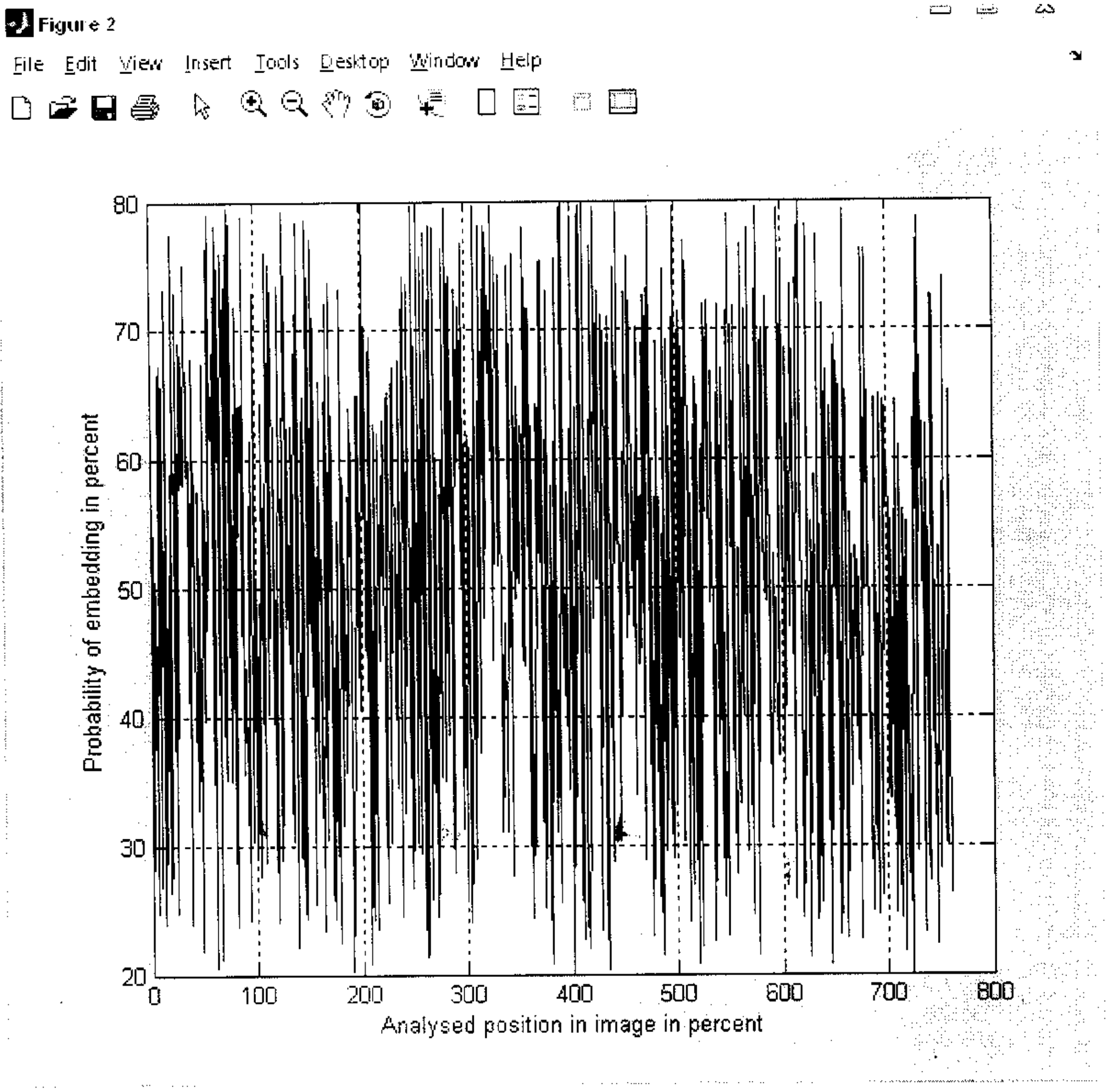
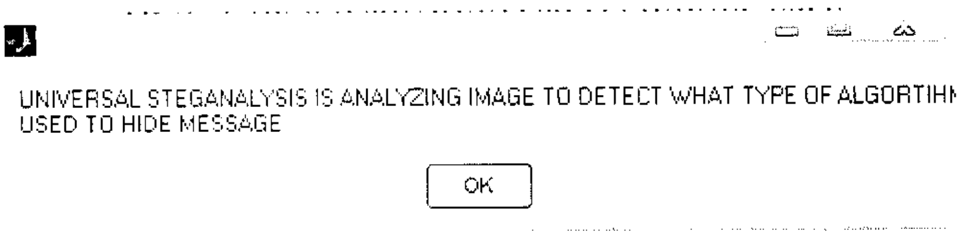
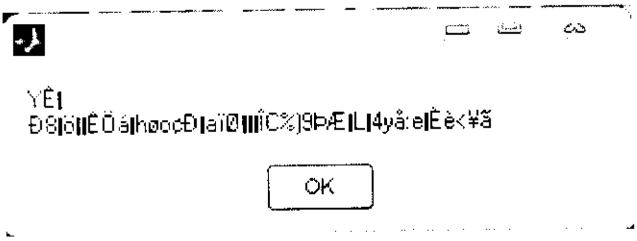


Fig A2.9: GRAPH SHOWING MIDDLE FREQUENCY TECHNIQUE USED



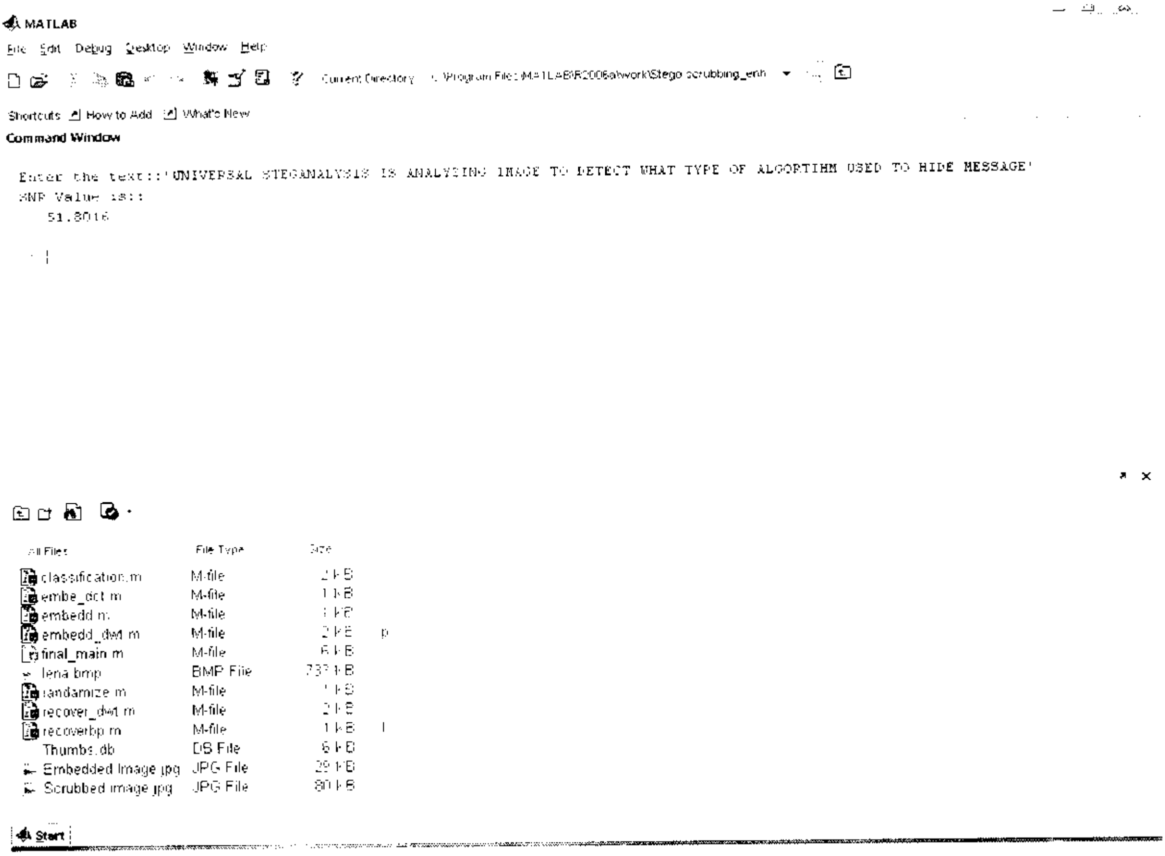
UNIVERSAL STEGANALYSIS IS ANALYZING IMAGE TO DETECT WHAT TYPE OF ALGORTIHI
USED TO HIDE MESSAGE

Fig A2.10: EXTRACTED MESSAGE BEFORE SCRUBBING



YÊI
ĐßlölÊÖálhøøcĐlaï@llllĈ%]9Þ/ÆlLl4yå:elÈè<¥ã

Fig A2.11: EXTRACTED MESSAGE AFTER SCRUBBING

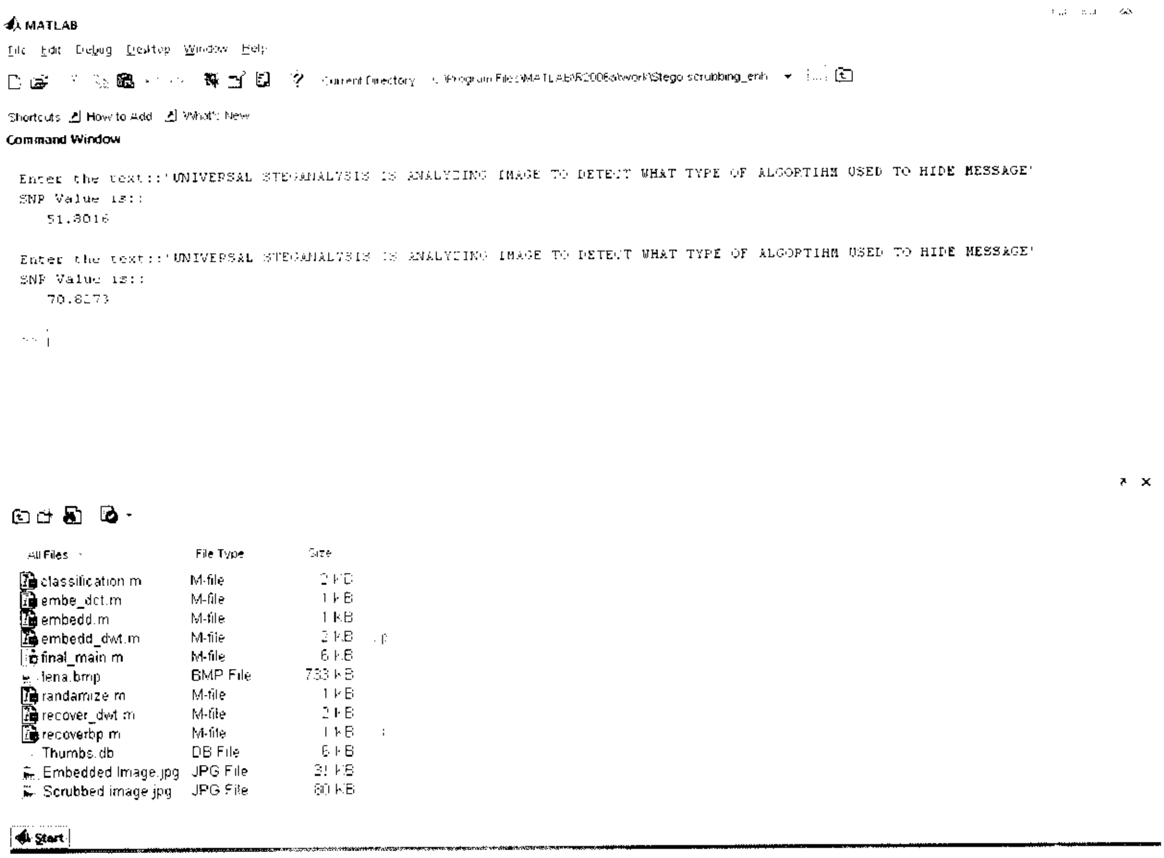


MATLAB
File Edit Debug Desktop Window Help
Current Directory C:\Program Files\MATLAB\R2006a\work\Stego scrubbing_enh
Shortcuts How to Add What's New
Command Window
Start

```
Enter the text::'UNIVERSAL STEGANALYSIS IS ANALYSING IMAGE TO DETECT WHAT TYPE OF ALGORTIHM USED TO HIDE MESSAGE'
SNR Value is::
    51.8016
```

All Files	File Type	Size
classification.m	M-file	2 KB
embe_dct.m	M-file	1 KB
embedd.m	M-file	1 KB
embedd_dwt.m	M-file	2 KB
final_main.m	M-file	6 KB
lena.bmp	BMP File	737 KB
randamize.m	M-file	1 KB
recover_dwt.m	M-file	2 KB
recoverbp.m	M-file	1 KB
Thumbs.db	DB File	6 KB
Embedded Image.jpg	JPG File	29 KB
Scrubbed image.jpg	JPG File	80 KB

Fig A2.12: QUALITY ASSESSMENT-DCT_MIDDLE FREQUENCY



MATLAB
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
Command Window
Start

Current Directory C:\Program Files\MATLAB\R2006a\work\Stego scrubbing_enh

```
Enter the text::'UNIVERSAL STEGANALYSIS IS ANALYZING IMAGE TO DETECT WHAT TYPE OF ALGORTIHM USED TO HIDE MESSAGE'
SNR Value is::
    51.8016

Enter the text::'UNIVERSAL STEGANALYSIS IS ANALYZING IMAGE TO DETECT WHAT TYPE OF ALGORTIHM USED TO HIDE MESSAGE'
SNR Value is::
    70.8173

>> |
```

All Files	File Type	Size
classification.m	M-file	2 KB
embe_dct.m	M-file	1 KB
embedd.m	M-file	1 KB
embedd_dwt.m	M-file	3 KB
final_main.m	M-file	6 KB
lena.bmp	BMP File	733 KB
randamize.m	M-file	1 KB
recover_dwt.m	M-file	2 KB
recoverbp.m	M-file	1 KB
Thumbs.db	DB File	6 KB
Embedded Image.jpg	JPG File	31 KB
Scrubbed image.jpg	JPG File	80 KB

Fig A2.13: QUALITY ASSESSMENT-DWT_LOW FREQUENCY

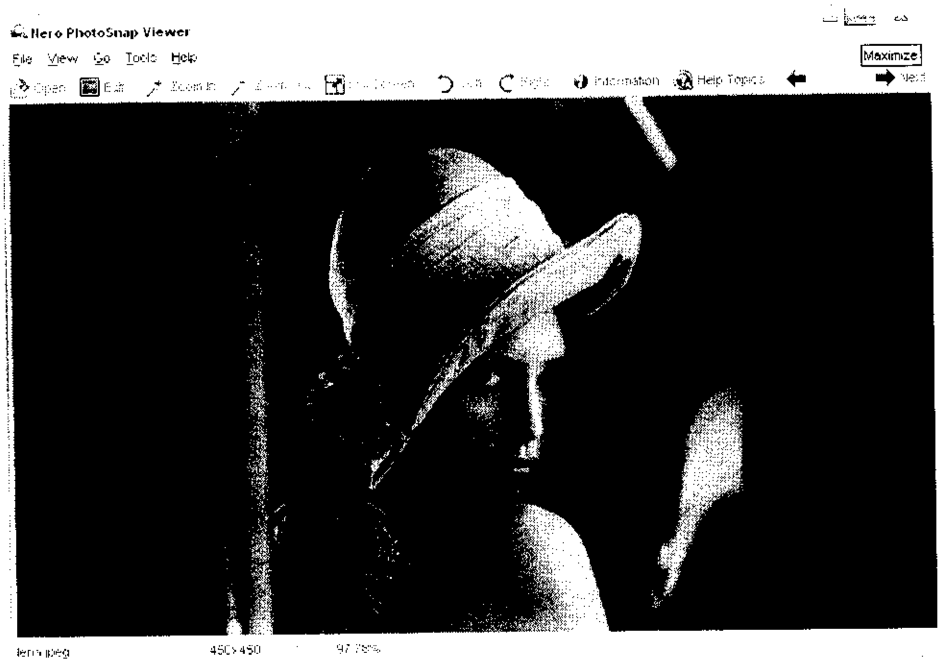

Nero PhotoSnap Viewer
Maximize
File View Go Tools Help
Open Edit Zoom In Zoom Out Full Screen Left Right Information Help Topics Next
lena.jpeg 450×450 97.78%

Fig A2.14: IMAGE BEFORE EMBEDDING MESSAGE

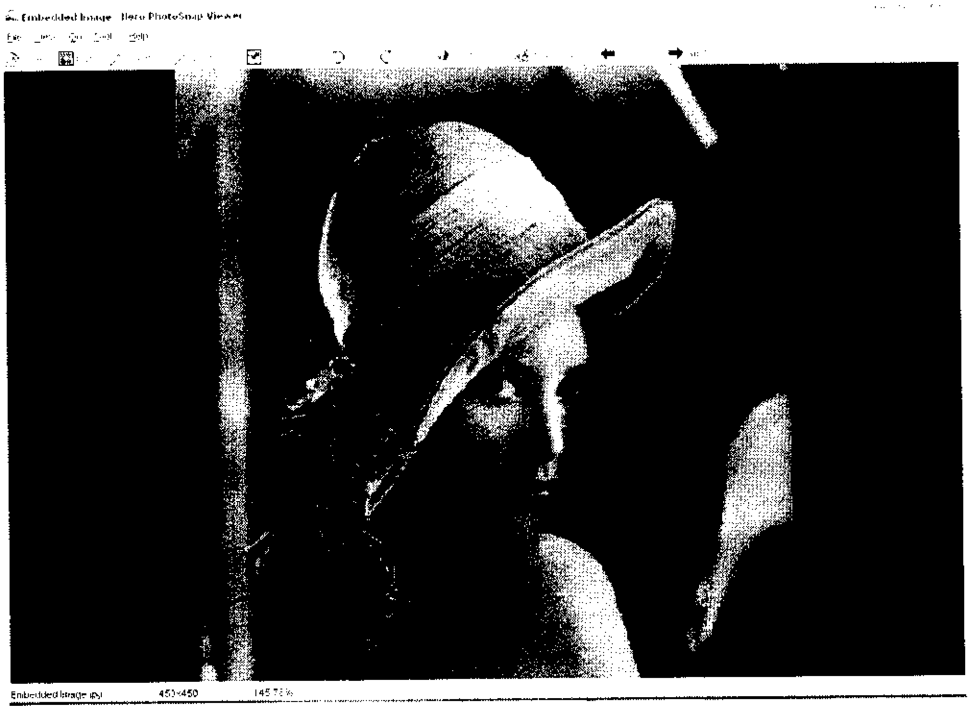

Embedded Image - Nero PhotoSnap Viewer
File View Go Tools Help
Embedded Image.jpg 450×450 145.78%

Fig A2.15: IMAGE WITH HIDDEN MESSAGE

REFERENCES

REFERENCES

[1].Donovan Artz & Z. Duric, "Digital Steganography", IEEE Internet Computing, May- June 2001.

[2]. Niels Provos &Peter Honeyman " Detecting Steganographic Content Internet " {provos,honey}@citi.umich.edu. August 31, 2001

[3]. N.F. Johnson & S. Jajodia, "Information Hiding: Steganography and Watermarking – Attacks and Counters – measures", June-July 2003.

[4]. J. J. Fridrich, M. Goljan, D. Hogea, and D. Soukal, "Quantitative steganalysis of digital images: estimating the secret message length.," Multimedia Syst., vol. 9, no. 3, pp. 288-302, 2003.

[5]. Petitcolas, Fabien A. P. "The information hiding homepage: digital watermarking & steganography." 17 Jun. 2002.

[6]. Anthony Whitehead Carleton University," Towards Eliminating Steganographic Communication", March 2005

[7]. K. Hempstalk, "Hiding behind corners: Using edges in images for better steganography," in Proceedings of the Computing Women's Congress 2006, Hamilton, New Zealand.