MAJOR THESIS REPORT

On

DCT BASED DIGITAL VIDEO WATERMARKING USING HUMAN VISUAL SYSTEM PROPERTIES

(Implementation and Verification by using MATLAB)

Submitted in the Partial Fulfilment of the Requirements for the Award of
The Degree of

MASTER OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION

By

GAURAV GUPTA **University Roll no. 2852**

Under the Guidance of

Mr. JEEBANANDA PANDA

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGG.
DELHI COLLEGE OF ENGINEERING, DELHI-110042
(SESSION 2004-2006)

CERTIFICATE

This is to certify that the MAJOR PROJECT REPORT titled as "DCT BASED VIDEO WATEMARKING USING HUMAN VISUAL SYSTEM PROPERTIES" (implementation and verification by using MATLAB) is being submitted by Gaurav Gupta, Class Roll No. 08/E&C/04, University Roll No. 2852, in partial fulfillment for the award of "Master of Engineering Degree in Electronics & Communication" in Delhi College of Engineering, Delhi University, Delhi; is the original work carried out by him under my guidance and supervision. The matter contained in this report has not been submitted elsewhere for award of any other degree.

PROJECT GUIDE

MR. JEEBANANDA PANDA
SENIOR LECTURER
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
DELHI COLLEGE OF ENGINEERING
BAWANA ROAD ,DELHI-42

ACKNOWLEDGEMENT

Firstly, I would like to express my heartily gratitude and thanks to my project guide Mr. Jeebananda Panda Lecturer in Deptt. of Electronics and Communication Engg; D.C.E. Delhi, for continuous inspiration, encouragement and guidance in every stage of preparation of this major thesis report.

I am also extremely thankful to Prof. Ashok Bhattacharyya, Head of the Deptt. of Electronics and Communication Engineering, D.C.E. Delhi, for the support provided by him during the entire duration of degree course and especially in this thesis.

I am also thankful to all teaching and non-teaching staff at DCE, and my fellows, who have helped me directly or indirectly in completion of this thesis report.

GAURAV GUPTA
(M.E. ELECTRONICS AND COMMUNICATION ENGG.)
COLLEGE ROLL NO. 08/E&C/04
DELHI UNIVERSITY ROLL NO. 2852

ABSTRACT

The major thesis work carried out in this report is based on video image-In-image watermarking algorithms which also take in to consideration the Human visual system Properties to find efficient locations in video frames for robust and imperceptible watermarks. Here a new HVS is defined which is a optimized global masking map for hiding watermark signals by combining the frequency masking, spatial masking, and motion masking effects of HVS. After getting adequate locations for insertion of watermark in parent video frames another video sequence is embedded frame by frame as watermark. For embedding the video sequence two DCT based algorithms are used which do embedding in spatial domain by modifying the value of selected coefficients. The transform operation is carried out on 4*4 blocks of each frame for DCT .Later video scene characteristic detection to improve digital watermarking transparency is carried out by subjective test analysis using different video sequences and video scene content characterstics were detected so that the best possible locations in a video scene to embed watermark are determined which the human eyes HVS is not easily able to detect ,which means give maximum imperceptibility.

CONTENTS

LIST OF FIGURES

LIST OF FIGURES GENERATED IN MATLAB IN SIMULATION

CHAPTER# 1

INTRODUCTION

1.1 INTRODUCTION

Steganography is derived from the Greek for covered writing and essentially means “to hide in plain sight”. As defined by Cachin [8] steganography is the art and science of communicating in such a way that the presence of a message cannot be detected. Simple steganographic techniques have been in use for hundreds of years, but with the increasing use of files in an electronic format new techniques for information hiding have become possible.

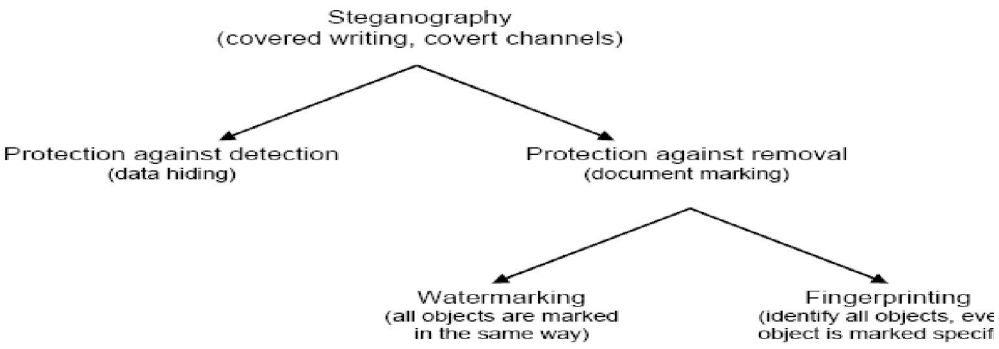


Figure 1.1 shows how information hiding can be broken down into different areas. Steganography can be used to hide a message intended for later retrieval by a specific individual or group. In this case the aim is to prevent the message being detected by any other party.

The other major area of steganography is copyright marking, where the message to be inserted is used to assert copyright over a document. This can be further divided into watermarking and fingerprinting which will be discussed later.

Steganography and encryption are both used to ensure data confidentiality. However the main difference between them is that with encryption anybody can see that both parties are communicating in secret. Steganography hides the existence of a secret message and in the best case nobody can see that both parties are communicating in secret.

This makes steganography suitable for some tasks for which encryption aren't, such as copyright marking. Adding encrypted copyright information to a file could be easy to remove but embedding it within the contents of the file itself can prevent it being easily identified and removed.

	Confidentiality	Integrity	Unremovability
Encryption	Yes	No	Yes
Digital Signatures	No	Yes	No
Steganography	Yes / No	Yes / No	Yes

Figure 1.2 shows a comparison of different techniques for communicating in secret. Encryption allows secure communication requiring a key to read the information. An attacker cannot remove the encryption but it is relatively easy to modify the file, making it unreadable for the intended recipient.

1.2 HISTORY

During the American Revolution, invisible ink which would glow over a flame was used by both the British and Americans to communicate secretly [8].

Steganography was also used in both World Wars. German spies hid text by using invisible ink to print small dots above or below letters and by changing the heights of letter-strokes in cover texts [9].

In World War I, prisoners of war would hide Morse code messages in letters home by using the dots and dashes on i, j, t and f. Censors intercepting the messages were often alerted by the phrasing and could change them in order to alter the message. A message reading “Father is dead” was modified to read “Father is deceased” and when the reply “Is Father dead or deceased?” came back the censor was alerted to the hidden message.

During World War II, the Germans would hide data as microdots. This involved photographing the message to be hidden and reducing the size so that that it could be used as a period within another document. FBI director J. Edgar Hoover described the use of microdots as "the enemy's masterpiece of espionage".

A message sent by a German spy during World War II read:

"Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects for pretext embargo on by-products, ejecting suets and vegetable oils."

By taking the second letter of every word the hidden message "Pershing sails for NY June 1" can be retrieved.

More recent cases of steganography include using special inks to write hidden messages on bank notes and also the entertainment industry using digital watermarking and fingerprinting of audio and video for copyright protection.

1.3 HIDING INFORMATION DIGITALLY

There are many different protocols and embedding techniques that enable us to hide data in a given object. However, all of the protocols and techniques must satisfy a number of requirements so that steganography can be applied correctly.

The following is a list of main requirements that steganography techniques must satisfy:

- The integrity of the hidden information after it has been embedded inside the stego object must be correct. The secret message must not change in any way, such as additional information being added, loss of information or changes to the secret information after it has been hidden. If secret information is changed during steganography, it would defeat the whole point of the process.
- The stego object must remain unchanged or almost unchanged to the naked eye. If the stego object changes significantly and can be noticed, a third party may see that information is being hidden and therefore could attempt to extract or to destroy it.
- In watermarking, changes in the stego object must have no effect on the watermark. Imagine if you had an illegal copy of an image that you would like to manipulate in various ways. These manipulations can be simple processes such as resizing, trimming or rotating the image. The watermark inside the image must survive these manipulations, otherwise the attackers can very easily remove the watermark and the point of steganography will be broken.

1.4 EMBEDING AND DETECTING WATERMARK

Figure 3 shows a simple representation of the generic embedding and decoding process in steganography. In this example, a secret image is being embedded inside a cover image to produce the stego image.

The first step in embedding and hiding information is to pass both the secret message and the cover message into the encoder. Inside the encoder, one or several protocols will be implemented to embed the secret information into the cover message. The type of protocol will depend on what information you are trying to embed and what you are embedding it in. For example, you will use an image protocol to embed information inside images.

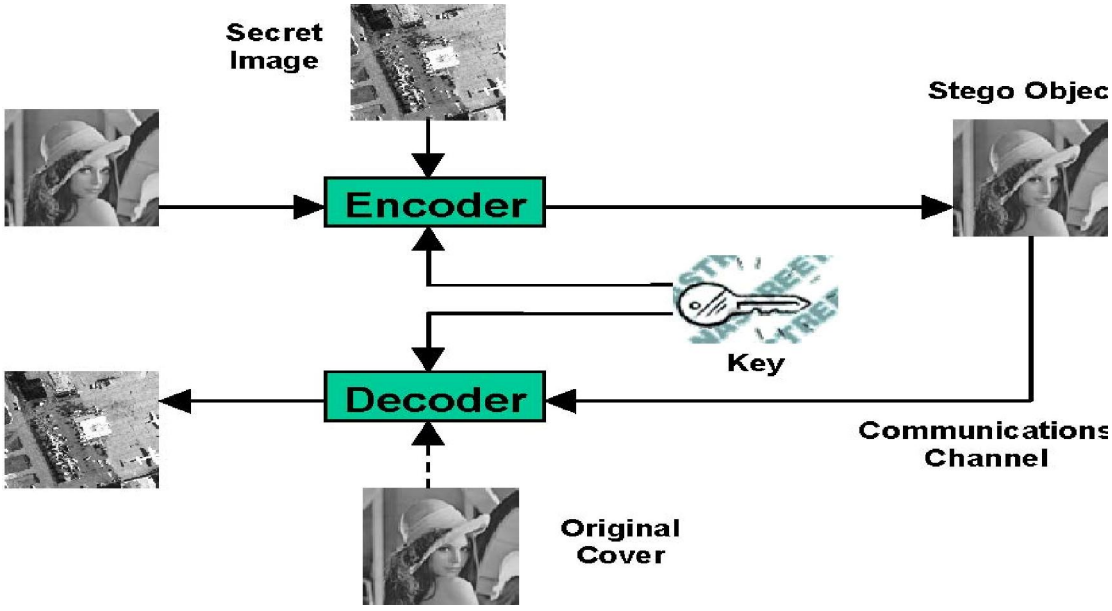


Figure 1.3. Generic process of encoding and decoding.

A key is often needed in the embedding process. This can be in the form of a public or private key so you can encode the secret message with your private key and the recipient can decode it using your public key. In embedding the information this way, you can reduce the chance of a third party attacker getting hold of the stego object and decoding it to find out the secret information.

Having passed through the encoder, a stego object will be produced. A stego object is the original cover object with the secret information embedded inside. This object should look almost identical to the cover object as otherwise a third party attacker can see embedded information.

Having produced the stego object, it will then be sent off via some communications channel, such as email, to the intended recipient for decoding. The recipient must decode the stego object in order for them to view the secret information. The decoding process is simply the reverse of the encoding process. It is the extraction of secret data from a stego object.

In the decoding process, the stego object is fed in to the system. The public or private key that can decode the original key that is used inside the encoding process is also needed so that the secret information can be decoded. Depending on the encoding technique, sometimes the original cover object is also needed in the decoding process. Otherwise, there may be no way of extracting the secret information from the stego object.

After the decoding process is completed, the secret information embedded in the stego object can then be extracted and viewed. The generic decoding process again requires a key, K, this time along with a potentially marked object, Ĩ'. Also required is either the mark, M, which is being checked for or the original object, I, and the result will be either the retrieved mark from the object or indication of the likelihood of M being present in Ĩ'.

1.5 TYPE OF STEGNOGRAPHY

Following section describes the definition of these two different types of steganography.

- Fragile

Fragile steganography involves embedding information into a file which is destroyed if the file is modified. This method is unsuitable for recording the copyright holder of the file since it can be so easily removed, but is useful in situations where it is important to prove that the file has not been tampered with, such as using a file as evidence in a court of law, since any tampering would have removed the watermark. Fragile steganography techniques tend to be easier to implement than robust methods.

- Robust

Robust marking aims to embed information into a file which cannot easily be destroyed. Although no mark is truly indestructible, a system can be considered robust if the amount of changes required to remove the mark would render the file useless. Therefore the mark should be hidden in a part of the file where its removal would be easily perceived.

There are two main types of robust marking. Fingerprinting involves hiding a unique identifier for the customer who originally acquired the file and therefore is allowed to use it. Should the file be found in the possession of somebody else, the copyright owner can use the fingerprint to identify which customer violated the license agreement by distributing a copy of the file.

Unlike fingerprints, watermarks identify the copyright owner of the file, not the customer. Whereas fingerprints are used to identify people who violate the license agreement watermarks help with prosecuting those who have an illegal copy. Ideally fingerprinting should be used but for mass production of CDs, DVDs, etc it is not feasible to give each disk a separate fingerprint.

Watermarks are typically hidden to prevent their detection and removal, they are said to be imperceptible watermarks.Visible watermarks can be used and often take the form of a visual pattern overlaid on an image. The use of visible watermarks is similar to the use of watermarks in non-digital formats. By taking advantage of human perception it is possible to embed data within a file. For example, with audio files frequency masking occurs when two tones with similar frequencies are played at the same time. The listener only hears the louder tone while the quieter one is masked. Similarly, temporal masking occurs when a low-level signal occurs immediately before or after a stronger one as it takes us time to adjust to the hearing the new frequency. This provides a clear point in the file in which to embed the mark.

1.6 STEGNOGRAPHY TECHNIQUES

However many of the formats used for digital media take advantage of compression standards such as MPEG to reduce file sizes by removing the parts which are not perceived by the users. Therefore the mark should be embedded in the perceptually most significant parts of the file to ensure it survives the compression process.

By taking advantage of human perception it is possible to embed data within a file. For example, with audio files frequency masking occurs when two tones with similar frequencies are played at the same time. The listener only hears the louder tone while the quieter one is masked. Similarly, temporal masking occurs when a low-level signal occurs immediately before or after a stronger one as it takes us time to adjust to the hearing the new frequency. This provides a clear point in the file in which to embed the mark.

Clearly embedding the mark in the significant parts of the file will result in a loss of quality since some of the information will be lost. A simple technique involves embedding the mark in the least significant bits which will minimize the distortion. However it also makes it relatively easy to locate and remove the mark. An improvement is to embed the mark only in the least significant bits of randomly chosen data within the file.

1.6.1 BINARY FILE TECHNIQUE

If we are trying to hide some secret information inside a binary file, whether the secret information is a copyright watermark or just simple secret text, we are faced with the problem that any changes to that binary file will cause the execution of it to alter. Just adding one single instruction will cause the executing to be different and therefore the program may not function properly and may crash the system.

It may be a wonder why people would want to embed information inside binary files, since there are so many other types of data format we can embed information in. The main reason for this is people want to protect their copyright inside a binary program. Of course there are other means of protecting copyright in software, such as serial keys, but if you did a search on the Internet, key generators for common programs are widely available and therefore using serial keys alone may not be enough to protect the binary file's copyright. One method for embedding a watermark in a binary file works as follows. First, let's look at the following lines of code that have been extracted from a binary file:

```
a = 2;
b = 3;
c = b + 3;
d = b + c;
```

The above instruction is simply equivalent to:

```
b = 3;      b = 3;          b = 3;
a = 2;      c = b + 3;      c = b + 3;
c = b + 3;  a = 2;          d = b + c;
d = b + c;  d = b + c;      a = 2;
```

The initialisation of b, c, and d must be done in the same order, but a can be initialised at any time.

To embed a watermark W = $\{w_1, w_2, w_3, w_4, \ldots, w_n\}$ where $w_i \in \{0, 1\}$. We first divide the source code into n blocks. Each of these blocks is then represented by w_i and this holds the value either 0 or 1. If w_i is 0, then the block of code it represents will be left unchanged. However, if w_i is 1, then you will look for two statements inside the block and switch them over.

Using this method, the watermark can be embedded by making changes to the binary code that does not affect the execution of the file. To decode and extract the watermark, you will need to have the original binary file. By comparing the marked and original files, you can then spot the statement switches and therefore extract the embedded watermark. This method is very simple but is not resistant to attacks. If the attacker has many different versions of the marked files then he may detect the watermark and hence be able to remove it.

1.6.2 TEST TECHNIQUE

While it is very easy to tell when you have committed a copyright infringement by photocopying a book, since the quality is widely different, it is more difficult when it comes to electronic versions of text.

Copies are identical and it is impossible to tell if it is an original or a copied version. To embed information inside a document we can

simply alter some of its characteristics. These can be either the text formatting or characteristics of the characters.

You may think that if we alter these characteristics it will become visible and obvious to third parties or attackers. The key to this problem is that we alter the document in a way that it is simply not visible to the human eye yet it is possible to decode it by computer.

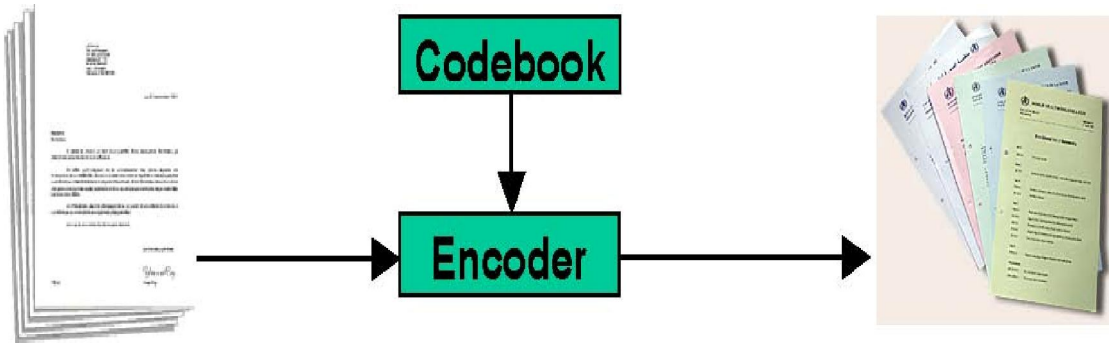


Figure 1.4. Document embedding process.

Figure 4 shows the general principle in embedding hidden information inside a document. Again, there is an encoder and to decode it, there will be a decoder. The codebook is a set of rules that tells the encoder which parts of the document it needs to change. It is also worth pointing out that the marked documents can be either identical or different. By different, we mean that the same watermark is marked on the document but different characteristics of each of the documents are changed.

1.6.2.1 Word Shift Coding Protocol

The word shift coding protocol is based on the same principle as the line shift coding protocol. The main difference is instead of shifting lines up or down, we shift words left or right. This is also known as the justification of the document.

The codebook will simply tell the encoder which of the words is to be shifted and whether it is a left or a right shift. Again, the decoding technique is measuring the spaces between each word and a left shift could represent a 0 bit and a right bit representing a 1 bit.

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

In this example the first line uses normal spacing while the second has had each word shifted left or right by 0.5 points in order to encode the sequence 01000001, that is 65, the ASCII character code for A. Without having the original for comparison it is likely that this may not be noticed and the shifting could be even smaller to make it less noticeable.

1.6.2.2 Feature Coding Protocol

In feature coding, there is a slight difference with the above protocols, and this is that the document is passed through a parser where it examines the document and it automatically builds a codebook specific to that document.

It will pick out all the features that it thinks it can use to hide information and each of these will be marked into the document. This can use a number of different characteristics such as the height of certain characters, the dots above i and j and the horizontal line length of letters such as f and t. Line shifting and word shifting techniques can also be used to increase the amount of data that can be hidden.

1.6.2.3 White Space Manipulation

One way of hiding data in text is to use white space. If done correctly, white space can be manipulated so that bits can be stored. This is done by adding a certain amount of white space to the end of lines. The amount of white space corresponds to a certain bit value.

Due to the fact that in practically all text editors, extra white space at the end of lines is skipped over, it won't be noticed by the casual viewer. In a large piece of text, this can result in enough room to hide a few lines of text or some secret codes. A program which uses this technique is SNOW [9], which is freely available.

1.6.2.4 Text Content

Another way of hiding information is to conceal it in what seems to be inconspicuous text. The grammar within the text can be used to store information. It is possible to change sentences to store information and keep the original meaning. TextHide [8] is a program, which incorporates this technique to hide secret messages.

A simple example is:

"The auto drives fast on a slippery road over the hill."

Changed to:

"Over the slope the car travels quickly on an ice-covered street."

Another way of using text itself is to use random words as a means of encoding information. Different words can be given different values.

Of course this would be easy to spot but there are clever implementations, such as SpamMimic [9] which creates a spam email that contains a secret message. As spam usually has poor grammar, it is far easier for it to escape notice. The following extract from a spam email encodes the phrase "I'm having a great time learning about computer security."

Dear Friend , Especially for you - this red-hot intelligence . We will comply with all removal requests . This mail is being sent in compliance with Senate bill 2116 , Title 9 ; Section 303 ! THIS IS NOT A GET RICH SCHEME . Why work for somebody else when you can become rich inside 57 weeks . Have you ever noticed most everyone has a cellphone & people love convenience . Well, now is your chance to capitalize on this . WE will help YOU SELL MORE and sell more ! You are guaranteed to succeed because we take all the risk ! But don't believe us . Ms Simpson of Washington tried us and says "My only problem now is where to park all my cars" . This offer is 100% legal . You will blame yourself forever if you don't order now ! Sign up a friend and you'll get a discount of 50% . Thank-you for your serious consideration of our offer . Dear Decision maker ;
Thank-you for your interest in our briefing . If you are not interested in our publications and wish to be removed from our lists, simply do NOT respond and ignore this mail ! This mail is being sent in compliance with Senate bill 1623 ; Title 6 ; Section 304 ! THIS IS NOT A GET RICH SCHEME ! Why work for somebody else when you can …

A very basic form of steganography makes use of a cipher. A cipher is basically a key which can be used to decode some data to retrieve a secret hidden message.

Sir Francis Bacon created one in the 16th Century [9] using messages with two different type faces, one bolder than the other. By looking at the positions of the bold characters in relation to the rest of the text, a secret message could be decoded. There are many other different ciphers which could be used to the same effect.

1.6.3 IMAGE TECHNIQUE

1.6.3.1 Simple Watermarking

A very simple yet widely used technique for watermarking images is to add a pattern on top of an existing image. Usually this pattern is an image itself - a logo or something similar, which distorts the underlying image.

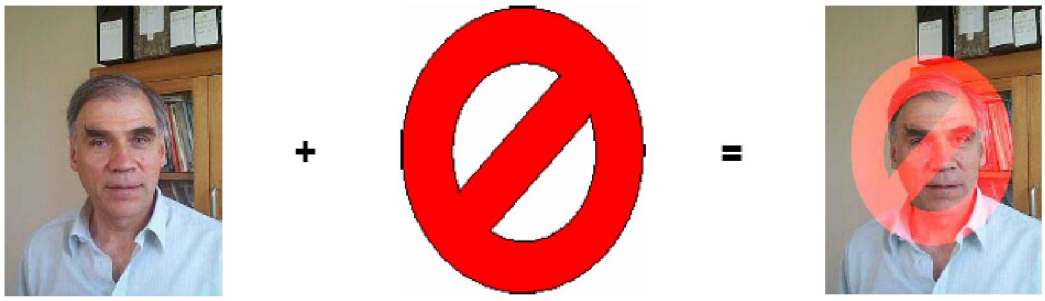


Figure 5. Visible watermarking.

In the example fig 1.5 above, the pattern is the red middle image while the portrait picture of Dr. Axford is the image being watermarked. In a standard image editor it is possible to merge both images and get a watermarked image. As long as you know the watermark, it is possible to reverse any adverse effects so that the original doesn't need to be kept.

This method is only really applicable to watermarking, as the pattern is visible and even without the original watermark, it is possible to remove the pattern from the watermarked image with some effort and skill.

1.6.3.2 LSB – Least Significant Bit Hiding (Image Hiding)

This method is probably the easiest way of hiding information in an image and yet it is surprisingly effective. It works by using the least significant bits of each pixel in one image to hide the most significant bits of another. So in a JPEG image for example, the following steps would need to be taken.

1. First load up both the host image and the image you need to hide.

2. Next chose the number of bits you wish to hide the secret image in. The more bits used in the host image, the more it deteriorates. Increasing the number of bits used though obviously has a beneficial reaction on the secret image increasing its clarity.

3. Now you have to create a new image by combining the pixels from both images. If you decide for example, to use 4 bits to hide the secret image, there will be four bits left for the host image. (PGM - one byte per pixel, JPEG - one byte each for red, green, blue and one byte for alpha channel in some image types)

Host Pixel: 1011<u>0001</u>

Secret Pixel: 0011<u>1111</u>

New Image Pixel: 10110011

4. To get the original image back you just need to know how many bits were used to store the secret image. You then scan through the host image, pick out the least significant bits according the number used and then use them to create a new image with one change - the bits extracted now become the most significant bits.

Host Pixel: 10110011

Bits used: 4

New Image: 00110000

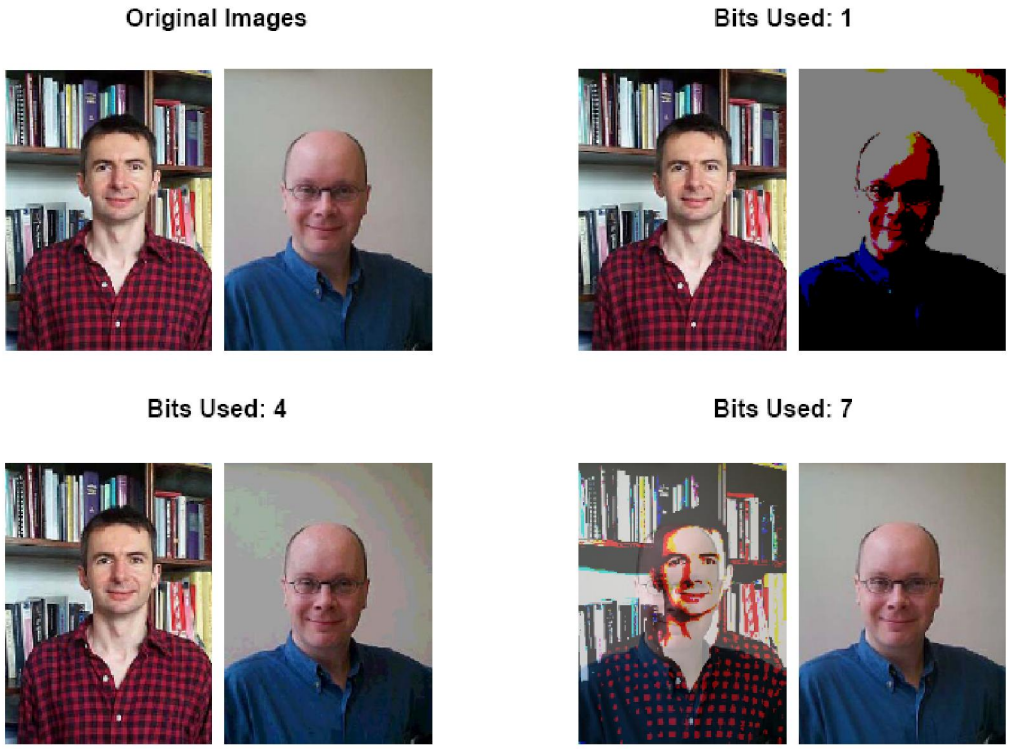


Figure 6. Least significant bit hiding.

To show how this technique affects images, Figure 1.6 shows examples using different bit values. Dr. Ryan's image on the left is the host image while Mr. Sexton's on the right is the secret one we wish to hide.

This method works well when both the host and secret images are given equal priority. When one has significantly more room than another, quality is sacrificed. Also while in this example an image has been hidden, the least significant bits could be used to store text or even a small amount of sound. All you need to do is change how the least significant bits are filled in the host image. However this technique makes it very easy to find and remove the hidden data [9].

1.6.3.3 Direct Cosine Transformation

Another way of hiding data is by way of a direct cosine transformation (DCT). The DCT algorithm is one of the main components of the JPEG compression technique [13]. This works as follows

1.First the image is split up into 8 x 8 squares.

2. Next each of these squares is transformed via a DCT, which outputs a multi dimensional array of 63 coefficients.

3.A quantizer rounds each of these coefficients, which essentially is the compression stage as this is where data is lost.

4.Small unimportant coefficients are rounded to 0 while larger ones lose some of their precision.

5.At this stage you should have an array of streamlined coefficients, which are further compressed via a Huffman encoding scheme or similar.

6.Decompression is done via an inverse DCT.

Hiding via a DCT is useful as someone who just looks at the pixel values of the image would be unaware that anything is amiss. Also the hidden data can be distributed more evenly over the whole image in such a way as to make it more robust.

One technique hides data in the quantizer stage [14]. If you wish to encode the bit value 0 in a specific 8 x 8 square of pixels, you can do this by making sure all the coefficients are even, for example by tweaking them. Bit value 1 can be stored by tweaking the coefficients so that they are odd. In this way a large image can store some data that is quite difficult to detect in comparison to the LSB method.

Figure 7. Direct Cosine Transformation.

FIG 1.7

Other techniques, which use DCT transformations, sometimes use different algorithms for storing the bit. One uses pseudo noise to add a watermark to the DCT coefficients while another uses an algorithm to encode and extract a bit from them.

1.6.3.4 Wavelet Transformation

While DCT transformations help hide watermark information or general data, they don't do a great job at higher compression levels.

The blocky look of highly compressed JPEG files is due to the 8 x 8 blocks used in the transformation process. Wavelet transformations on the other hand are far better at high compression levels and thus increase the level of robustness of the information that is hidden, something which is essential in an area like watermarking.

This technique works by taking many wavelets to encode a whole image. They allow images to be compressed so highly by storing the high frequency “detail” in the image separately from the low frequency parts. The low frequency areas can then be compressed which is acceptable as they are most viable for compression. Quantization can then take place to compress things further and the whole process can start again if needed.

A simple technique using wavelets to hide information is exactly like one of the techniques discussed in the previous section. Instead of altering the DCT coefficients with pseudo noise, instead the coefficients of the wavelets are altered with the noise within tolerable levels.

1.6.4 SOUND TECHNIQUE

1.6.4.1 Spread Spectrum

Spread spectrum systems encode data as a binary sequence which sounds like noise but which can be recognised by a receiver with the correct key.

The technique has been used by the military since the 1940s because the signals are hard to jam or intercept as they are lost in the background noise. Spread spectrum techniques can be used for watermarking by matching the narrow bandwidth of the embedded data to the large bandwidth of the medium.

1.6.4.2 MP3

The MP3 format is probably the most widespread compression format currently used for music files. Due to this, it also happens to be very good for hiding information in. The more inconspicuous the format, the more easily the hidden data may be overlooked.

There are very few working examples of hiding information in MP3 files but one freely available program is MP3 Stego . The technique used here is similar to the frequency transformations discussed earlier. Basically the data to be hidden is stored as the MP3 file is created, that is during the compression stage .

As the sound file is being compressed during the Layer 3 encoding process, data is selectively lost depending on the bit rate the user has specified. The hidden data is encoded in the parity bit of this information. As MP3 files are split up into a number of frames each with their own parity bit, a reasonable amount of information can be stored. To retrieve the data all you need to do is uncompress the MP3 file and read the parity bits as this process is done. This is an effective technique which leaves little trace of any distortions in the music file.

1.6.4.3 VIDEO

For video, a combination of sound and image techniques can be used. This is due to the fact that video generally has separate inner files for the video (consisting of many images) and the sound. So techniques can be applied in both areas to hide data. Due to the size of video files, the scope for adding lots of data is much greater and therefore the chances of hidden data being detected is quite low.

1.6.4.4 **DNA**

A relatively new area for information hiding is within DNA. In one technique explained by Peterson a message "JUNE6_INVASION:NORMANDY" was hidden inside some DNA. This was done in a scheme quite similar to some of the text techniques discussed earlier.

A single strand of DNA consists of a chain of simple molecules called bases, which protrude from a sugar-phosphate backbone. The four varieties of bases are known as adenine (A), thymine (T), guanine (G), and cytosine (C). A table was drawn up with different three base combinations equalling different words in the alphabet along with a few other things.

To create the secret message, DNA was synthesised following this table with the bases in the right order. Then it was sandwiched between another two strands of DNA which acted as markers to point the sender and recipient of the message to the message. The final step taken was to add in some random DNA strands in order to further prevent the detection of the secret message.

As DNA is incredibly small, it can be hidden in a dot in a book or magazine much like the old microdot technique used in World War II. It is also robust enough to be posted through the mail and still be decoded. This could prove to be a very effective technique in the future.

1.7 DETECTION

Although many of the uses of steganography are perfectly legal, it can be abused by certain groups. The potential exists for terrorist groups to communicate using these techniques to hide their messages and rumours persist that Al-Qaeda have used it to communicate. Also of concern is that these techniques may be used by paedophiles to hide pornographic images within seemingly innocuous material.

As a result the need for detection of steganographic data has become an important issue for law enforcement agencies. Attempting to detect the use of steganography is called steganalysis and can be either passive, where the presence of the hidden data is detected, or active, where an attempt is made to retrieve the hidden data.

This detection is similar to that described earlier for checking for the presence of a watermark. However, whereas before detection will be used when a mark is expected and may involve using the original file, in this case the original file is unavailable and there is no expected mark. Instead the file must be checked for the presence of data hidden in a variety of formats.

1.8 ATTACKS

Information hiding techniques still suffer from several limitations leaving them open to attack and robustness criteria vary between different techniques. Attacks can be broadly categorized although some attacks will fit into multiple categories.

1.8.1 Basic Attacks

Basic attacks take advantage of limitations in the design of the embedding techniques. Simple spread spectrum techniques, for example, are able to survive amplitude distortion and noise addition but are vulnerable to timing errors. Synchronization of the chip signal is required in order for the technique to work so adjusting the synchronization can cause t he embedded data to be lost.

It is possible to alter the length of a piece of audio without changing the pitch and this can also be an effective attack on audio files.

1.8.2 Robustness Attacks

Robustness attacks attempt to diminish or remove the presence of a watermark. Although most techniques can survive a variety of transformations, compression, noise addition, etc they do not cope so easily with combinations of them or with random geometric distortions. If a series of minor distortions are applied the watermark can be lost while the image remains largely unchanged. What changes have been made will likely be acceptable to pirates who do not usually require high quality copies. Since robustness attacks involve the use of common manipulations, they need not always be malicious but could just be the result of normal usage by licensed users.

Protecting against these attacks can be done by anticipating which transformations pirates are likely to use. Embedding multiple copies of the mark using inverse transformations can increase the resistance to these attacks.

The echo hiding technique encodes zeros and ones by adding echo signals distinguished by different values for their delay and amplitude to an audio signal. Decoding can be done by detecting the initial delay using the auto-correlation of the cepstrum of the encoded signal but this technique can also be used as an attack.

If the echo can be detected then it can be removed by inverting the formula used to add it. The difficult part is detecting the echo without any knowledge of the original or the echo parameters. This problem is known as ‘blind echo cancellation’. Finding the echo can be done using a technique called cepstrum analysis.

Other attacks will attempt to identify the watermark and then remove it. This technique is particularly applicable if the marking process leaves clues that help the attacker gain information about the mark. For example an image with a low number of colours, such as a cartoon image, will have sharp peaks in the colour histogram. Some marking algorithms split these and the twin peaks attack takes advantage of this to identify the marks which can then be removed.

CHAPTER # 2

ANALYSIS OF DCT AS PART OF IMAGE COMPRESSION AND FREQUENCY DOMAIN WATERMARKING ALGORITHMS

2.1 AMPLE SPACE IN COLOR MODELS

In order to combat the effects of image compression on our embedded message, we must gain an understanding of several key steps in the JPEG algorithm. These steps include converting from the RGB to YUV color space and transforming this information from the spatial domain to the frequency domain. The transformation to the frequency domain is done through the Discrete Cosine Transform (DCT), which has several key properties which can be used to further increase the robustness of our embedding method. Once these steps can be understood, we can increase the robustness of our embedding method by taking advantage of specific properties of each step.

Images are represented as a collection of individual elements called pixels. A pixel is a small dot on the screen, and represents a single color. Computer monitors display images by using three different phosphors (red, green, blue) for each pixel. A computer displays images in an additive manner, combining the three phosphors in order to generate a certain shade. Thus if all three phosphors are minimized, the pixel appears to be black. Likewise, if all three phosphors are maximized, the pixel is white. Different combinations of the three phosphors produce most of the colors in between. Most images are represented by the RGB (red-green-blue) color model, which mimics the way computer monitors work. Each pixel in the RGB color model is represented as a combination of these three components. The images dealt with in this thesis were 24 bit JPEGs, 8 bits for each component. The values of each component range from 0 to 255.

The JPEG file format converts from RGB to the YUV (luminance-chrominance-chrominance) color model. Specifically, it converts an image to a form of the YUV color model known as YCbCr. The YUV color model was developed as a way to allow television stations to broadcast in both black-and-white and color. Black-and-white television sets use only the luminance (Y) portion of the broadcast, ignoring the chrominance portions of the signal. The Y component specifies the intensity of the image. Color television sets also decode the two chrominance components (Cb,Cr) in order to form a color picture . The Cb component specifies how much blue is in the image, while the Cr component specifies how much red is in the image. The RGB and YUV colors models are related via the following formulas:

$$Y = 0.299R + 0.587G + 0.114B$$
$$U = -0.1687R - 0.3313G + .5B + 2^{\text{Sample Precision}/2}$$
$$V = 0.5R - 0.4187G - 0.0813B + 2^{\text{Sample Precision}/2}$$

$$R = Y + 1.402(V - 2^{\text{Sample Precision}/2})$$
$$G = Y - 0.34414(U - 2^{\text{Sample Precision}/2}) - 0.71414(V - 2^{\text{Sample Precision}/2})$$
$$B = Y + 1.722(U - 2^{\text{Sample Precision}/2})$$

Figure 2.1: RGB to YUV (YCrCb) conversion formulas [2]. Sample Precision is equal to 8 for JPEG images[1]

The luminance component (Y) contains a large part of the color information of the RGB color model, as viewed in the formula which converts YUV back to RGB. This is made obvious in Figure 2.3, which shows an image of a tiger broken up into its Y, U, and V components. This is an important property of the YUV color model. It allows compression to be performed on the chrominance (U,V) components of an image, without affecting our visual perception of it. This is very different from the RGB color model, which assigns all three components equal importance.

Compression algorithm exploits this fact, performing greater compression on the chrominance components than the luminance component. Data hidden in the luminance component of an image will be hidden in the *more-significant* part of the data, and less likely to be damaged during compression. By exploiting this characteristic of the YUV color model, we attempt to make our stego embedding process more robust.

2.2 FREQUENCY DOMAIN FOR EMBEDDING WATERMARK

Another step we can take to make the embedding process more robust is by transforming the data from the YUV color space into the frequency domain. This allows us to look at images as a continuous entity, rather than a discrete set of 1‘s and 0‘s. Bits can now be embedded according to thresholds, rather than embedding the exact bit. In doing so, we are given greater freedom in embedding data, and greater margin for error caused by compression. The frequency domain represents the amount of energy in the luminance and chrominance light waves. The energy distribution varies greatly depending upon an image, and so it is difficult to compress an image in the spatial domain. In the frequency domain, the energy in an image tends to be more compact.

Figure 2.2: Original color image of a tiger (photo by Josh Buchanan)

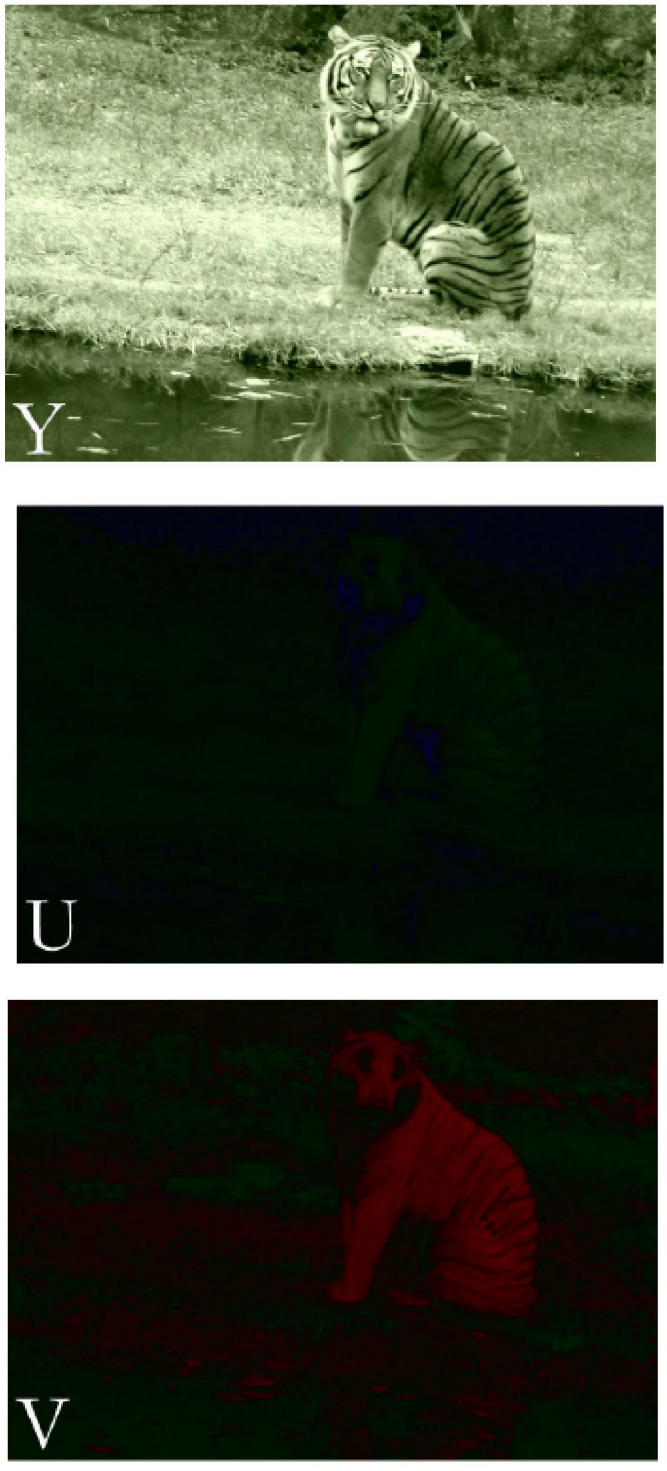


Figure 2.3: Image of a tiger, separated into Y, U, and V components.
(photo by Josh Buchanan)

Grouped closer together. Thus the JPEG compression algorithm transforms image data into the frequency domain, which allows compression to be performed on the outer components of the energy spectrum.

2.3 DCT TRANSFORMATION

The operation that JPEG uses to convert between the spatial domain and the frequency domain is the two-dimensional Discrete Cosine Transform (DCT). The JPEG compression algorithm partitions an image into 8x8 blocks of pixels, applying the DCT separately to each block [18]. The DCT is a singular transform which transforms data into a sum of cosine functions. The two-dimensional DCT is defined as follows:

$$DCT_{uv} = \frac{1}{\sqrt{2N}} C_u C_v \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} DCT_{ij} \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)u\pi}{2N}\right]$$

$$\text{where } C_u, C_v = \begin{cases} 1/\sqrt{2} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

Figure 2.4: Formula for the 2-D Discrete Cosine Transform (DCT). In JPEG compression, N=8.

So why use the DCT over the Discrete Fourier Transform (DFT)? This equation represents a special derivation of the Fourier transform. The Fourier Transform treats a finite signal as if it were periodic. Thus a difference in the edges of two neighboring blocks in an image results in high frequency components. During the quantization step (see Section 2.5), these high frequencies are eliminated, resulting in blocky artifacts in the image. Thus the DFT needs to be modified in order to work well with images.

The Fourier series of any continuous, real-valued symmetric function is composed entirely of coefficients relating to the cosine terms of the series. By forcing symmetry on the DFT, we are able to utilize this property to produce results which are real-valued and can be expressed entirely in terms of cosine waves . This has an important consequence. The symmetry of the function removes the sharp change between neighboring image blocks, eliminating the high-frequency components that cause blocky artifacts in an image [9].

2.4 ANALYSIS BY EXAMPLE

Let‘s observe the basic behavior of the DCT transform by viewing a sample calculation from an 8x8 block of the eye of the tiger in the previous section. Figure 2.5 shows the Y values for the image, and the corresponding DCT coefficients. The upper left-hand value in the DCT matrix (921) is a constant value, known as the DC component.

The other values in the matrix are known as AC components. The DCT separates an image into parts according to its visual importance. That is, it transforms most of the wave energy into the upper left corner of the block of coefficients, and the least energy into the bottom right corner of the block. Thus the DC component represents the most-significant part of the data. You can see this in the example calculation, as the DC component is much larger than any of the other coefficients. As you move towards the right-hand corner of the matrix, the magnitude of the values gets increasingly smaller. It is excellent at energy compaction, and it is important to note that this property is independent of the image data. As seen in Figure 2.6, most of the energy of the flower image is also concentrated in the upper left hand corner of the coefficient matrix. This is an important part of the JPEG compression scheme. The algorithm is able to compress the lower right hand corner without affecting the visual perception of the image.

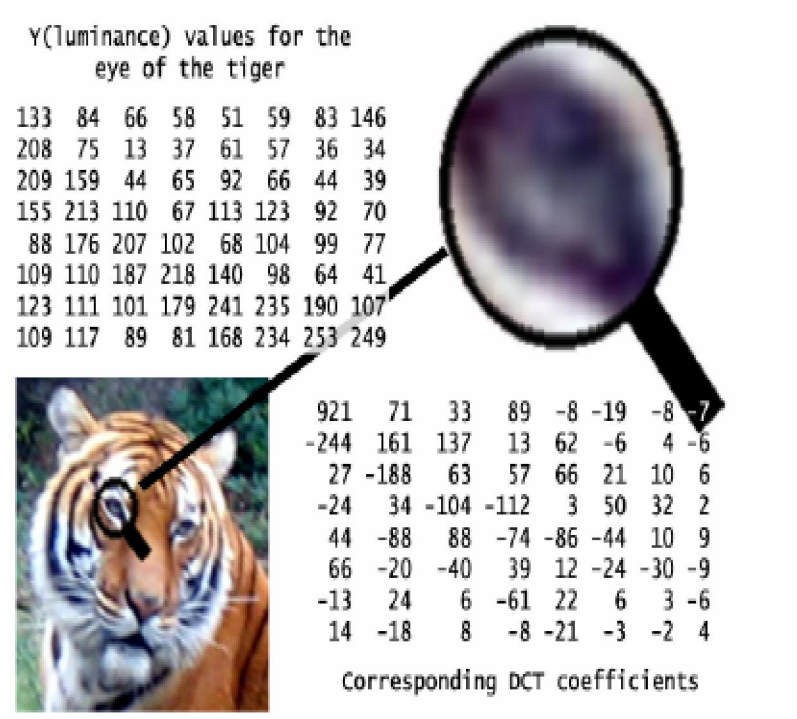


Figure 2.5: An example 8x8 block from the tiger image and its DCT coefficients. Coefficients were calculated using Matlab.

It is important to apply the notions of significant bits in the spatial domain that were introduced in the first chapter to the DCT coefficients in the frequency domain. Using the DCT, the DC coefficient is analogous to the most-significant bit in the spatial domain. Similarly, the least-significant bits are akin to the coefficients represented in the lower right-hand corner. In our search for the *more-significant* bits of data, we look to the middle frequency DCT coefficients. These coefficients are somewhat arbitrary, and will be obtained through research and experimental data.

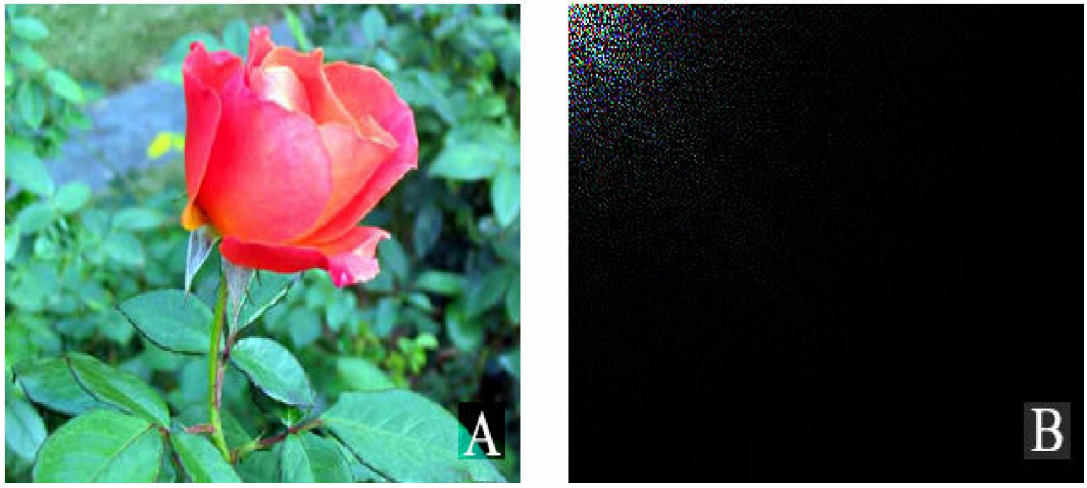


FIG 2.6 DCT of an image

2.5 QUANTIZATION EFFECT

Up until now, the steps of the JPEG compression algorithm have been lossless. The lossy part of the algorithm takes place during the quantization of each 8x8 image block, and is the part our information hiding process attempts to withstand. Quantization on a matrix *M* is performed as follows:

$$M_Q[x, y] = \text{Round}\left(\frac{M[x, y]}{q[x, y]}\right)$$

Here *MQ* represents the quantized matrix, and *q* is the matrix representing the quantization table. The quantization table is an 8x8 matrix, and is stored in the header of the JPEG file format. Each value in the original matrix *M* has a corresponding value in *q*. Higher values in the quantization table result in more round-off in the quantized matrix *MQ*. This fact is demonstrated in the image restoration process, which occurs as follows: $M[x,y]=M_Q[x,y]*q[x,y]$

Again *MQ* represents the quantized matrix, *q* is the matrix representing the quantization table, and *M* is the restored image. Initially, this inverse operation may seem trivial. However, compression is achieved through the initial rounding of the coefficient, the structure of the quantization table, and the nature of the D.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figure 2.7 Quantization Table for the Y Component [2]

The JPEG committee does not specify an exact standard for quantization tables, but it does provide a table (Figure 2.7) that has been tested with good results. Notice how the quantization values corresponding to the lower right-hand corner of the DCT coefficients are larger than the others. This takes advantage of the fact that the DCT concentrates most of its energy into the upper left hand corner of the matrix. Figure 2.8 shows the effects of quantizing on the 8x8 block representing the tiger eye. Most of the visually less-significant coefficients are rounded down to 0. Thus when these coefficients are restored, they will be restored as 0. The JPEG algorithm then applies Huffman encoding to the matrix in a zigzag manner in order to achieve maximum compression on these 0‘s.

So how does quantization affect the stego algorithm? The method must be robust enough to survive the quantization stage of JPEG compression. Since there is no set standard for quantization tables, it must be able to survive a variety of tables.

58	6	3	6	0	0	0	0
-20	13	10	1	2	0	0	0
2	-14	4	2	2	0	0	0
-2	2	-5	-4	0	1	0	0
2	-4	2	-1	-1	0	0	0
3	-1	-1	1	0	0	0	0
0	0	0	-1	0	0	0	0
0	0	0	0	0	0	0	0

Figure 2.8: Results of Quantizing the DCT coefficients from Figure 2.5

This chapter introduced steps of the JPEG algorithm which will affect the robustness of the proposed stego embedding process. The luminance (Y) component has been identified as the visually most-significant region of the YUV color space, and thus the component that is least likely to be compressed. This component is where data will be embedded. We examined the DCT and its basic properties, and observed how it distributed most of the energy from the spatial domain into the upper left hand corner of the coefficient matrix. Finally, we looked at how data is compressed using quantization tables, and how the layout of these tables takes advantage of the compaction properties of the DCT.

CHAPTER # 3

THE HUMAN VISUAL SYSTEM

3.1 INTRODUCTION

The first problem that all data-embedding and watermarking schemes need to address is that of inserting data in the digital signal without deteriorating its perceptual quality. We must be able to retrieve the data from the edited host signal. Because the data insertion and data recovery procedures are intimately related, the insertion scheme must take into account the requirement of the data-embedding applications. Data insertion is possible because the digital medium is ultimately consumed by a human. The human hearing and_visual systems are imperfect detectors. Audio and visual signals must have a minimum intensity or contrast level before they can be detected by a human. These minimum levels depend on the spatial, temporal and frequency characteristics of the human auditory and visual systems. Most signal-coding techniques exploit the characteristics of the human auditory and visual systems directly or indirectly. Likewise, all data-embedding techniques exploit the characteristics of the human auditory and visual systems implicitly or explicitly. A diagram of a data-embedding algorithm is shown in Figure 3.1. The information is embedded into the signal using the embedding algorithm and a key. The dashed lines indicate that the algorithm may directly exploit perceptual analysis to embed information. In fact, embedding data would not be possible without the limitations of the human visual and auditory systems.

Data embedding and watermarking algorithms embed text, binary streams, audio, image or video in a host audio, image or video signal. The embedded data are perceptually inaudible or invisible to maintain the quality of the source data.

We are going to use a new video watermarking algorithm based on the human visual system (HVS) properties to find effective locations in video sequences for robust and imperceptible watermarks. In particular, we define a new HVS-optimized global masking map for hiding watermark signals by combining the spatial masking, and the motion masking effects of HVS.

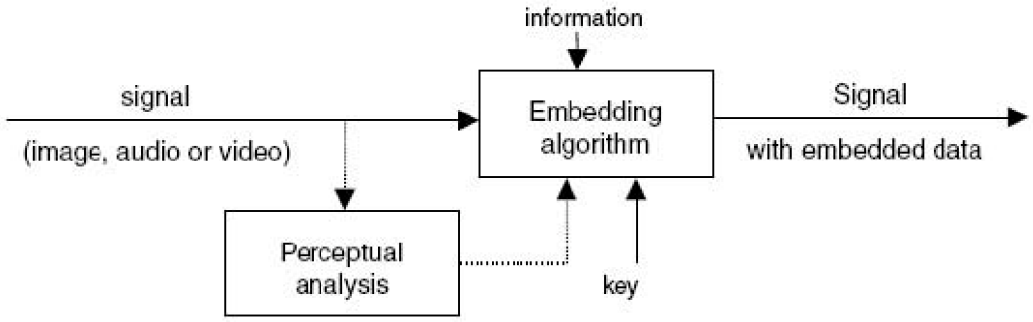


FIG 3.1 Perceptual analysis

3.2 THE GLOBAL MASKING MAP

3.2.1 Frequency Masking

The frequency masking operation is based on the Watson's visual model, defined in the DCT domain . After we divide the image *I* into 4x4 pixel l blocks $I[x,y]$ k, where $x,y = 0,1,\ldots,3$ and k is the block number, each block is transformed into the DCT domain. Then, we use the frequency sensitivity table, which contains the smallest magnitude of each DCTcoefficient in the block that is perceptible in the absence of any masking noise.

Thus, as smaller value in the table indicates that the eye is more sensitive to the frequency component.

In the absence of masking noise in the frequency domain, we can find a visibility threshold value that is the minimum level below which the signal is not perceptible .The visibility threshold is defined by

$$t_{th}[i,j]_k = t[i,j][\frac{\tilde{I}[0,0]_k}{\tilde{I}_{avg}[0,0]}]^{\alpha} \quad (1)$$

where $\tilde{I}[0,0]k$ is the DC coefficient of the k^{th} block in the original image, $\tilde{I}_{avg}[0,0]$ is the average of the DC coefficients in the image, and a is a constant. In our experiment, we set a=0.649 empirically. The frequency masking function is also defined by

$$\tilde{F}[i,j]_k = t_{th}[i,j] \cdot max\{1, [\frac{|\tilde{I}[i,j]_k|}{\tilde{I}_{th}[i,j]_k}]^{w}\} \quad (2)$$

where $\tilde{I}[i,j]$ K is the DCT coefficients of the k^{th} block and w is a constant.

3.2.2 Spatial Masking

The main purpose of the edge map in the propose d watermarking algorithm is to extract connected edges in each image frame. Here, we control the contrast of images before the edge detection operation to obtain a good spatial masking map from the following lightness function.

Where $I[x,y]$ is the luminance value of the original frame. Schreiber indicated that a=0.05 provides a well-adapted luminance scale. After we apply the lightness function. We extract important edges to fid the spatial masking effect.

$$S[x,y] = 1 + 99\frac{log(1 + I[x,y] \cdot a) - log(1 + a)}{log(1 + 100a) - log(1 + a)} \qquad (3)$$

3.2.3 Motion Masking

A video watermarking method can exploit the structural characteristics of the video sequence. After we find displacement parts in the successive image frames, we can apply a suitable filter to extract image contours. Figure (B) shows the result of the motion masking, where high values are assigned in the face part because of large motion changes.

(A)SPATIAL MASKING (B) MOTION MASKING

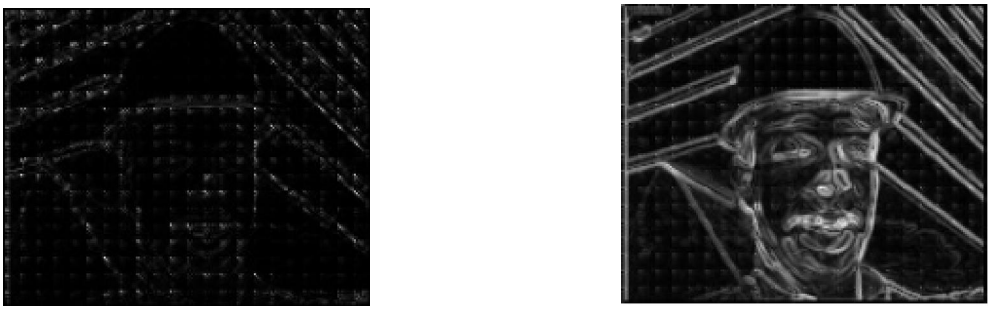
FIG 3.2

3.2.4 Global Masking Map Modeling

We define the global masking map by combining the above frequency, spatial, motion masking effects, we extract important edges by combining frequency, spatial, and motion masking effects together after normalization. In other words, the global masking map *G* is obtained by

$$G = F + S + M \qquad (4)$$

Where F is the frequency masking, S is the spatial masking, and M is the motion masking ,respectively. Figure3 shows simulation results with the global masking; Figure (a) is obtained by the previous method, and Figure (b) is obtained by the proposed method. We can see that Figure(b) is more effective than Figure(a); that is, we can insert more watermarks effectively using the proposed global masking map more than the previous one.



Global masking result: (a) The previous method (b) The proposed method

FIG 3.3

3.3 VIDEO WATERMARKING IMPERCEPTIBLITY IMPROVEMENT

Imperceptibility of watermarking method in terms of PSNR is generally high which is a objective criteria .However it should be noted that the PSNR only compares the watermarked frame with the original unwatermarked frame , hence there are possible temporal artifacts that cannot be measured by the PSNR .

In image watermarking ,the imperceptibility can be improved by embedding the watermark in the places where the human visual system is not too sensitive . we can exploit this by using texture and luminance masking .

By texture masking ,the watermark can be embedded more strongly in the blocks with the stronger texture .

Whereas by luminance masking the watermark can be embedded more strongly in the blocks with the brighter back ground.

If these masking factor are to be implemented in video we should be aware of any flickering effect and this artifact should be minimised .Based on the work of the physiology of eye the flicker threshold is related to 'critical fusion frequency' which is defined as the point where the flicker sensation disappears to be replaced by the sensation of continuous simulation.

Then the critical frequency is proportional to the logarithmic of the luminance of the flickering patch or practically the flicker can be reduced by decreasing the luminance intensity.

The above principle is employed to improve the imperceptibility of our video video watermarking scheme by embedding the watermark in these blocks ,the flickering effect might be decreased .

To test the validity of this approach subjective test analysis is done on eight different video sequences with four embedding conditions.

Low-texture blocks with any luminance(random)
#Low-texture blocks with low luminance
High-texture blocks with any luminance(random)
High-texture blocks with low luminance

3.4 VIDEO CONTENT DETECTION TO SUIT PARTICULAR EMBEDDING SCHEME

Unlike the image watermarking where the imperceptibility is high if the watermark is embedded in high texture part ,in video high imperceptibility is not always produced by high –texture embedding .The internal characteristic of the sequence plays an important role for choosing the embedding method if high video imperceptibility is required.
Intuitively we can assume that the movement part in a frame is more recognizable by human eye than any other static part or in other words the part of a frame which has high difference between its adjacent frames is not a good place to embed watermark.

There are three expects of a video scene that will be examined namely the texture the difference between adjacent frames and the luminance.

First for each frame

1.The low texture blocks T_L
2.The low texture blocks which have low difference between adjacent frames D_L are marked and counted so that for all n frames the proportion of low difference blocks within the low texture blocks can be calculated.

$$LDLTR = 1/n \sum_{i=1}^{n} D_L / T_L$$

3.low texture block luminances are measured so that average LTL can be measured.

$$LTC = LDLTR / LTL$$

The next step is to find the high texture characterstics .the characterstics HTC,is estimated by same procedure

1.The high texture blocks T_H

2.The high texture blocks which have low difference between adjacent frames D_H are marked and counted so that for all n frames the proportion of low difference blocks within the high texture blocks can be calculated.

$$LDHTR = 1/n \sum_{i=1}^{n} D_H / T_H$$

3. High texture block luminances are measured so that average HTL can be measured.

HTC = LDHTR / HTL

By comparing these low-texture and high texture characteristics and settings a threshold the dominant characteristic for every scene can be determined and a particular embedding method can be choosen.

The approach described above is being implemented on the eight different sequences that were used in the exeperiment .The video scene characterstic described above is applicable to a single video scene .

Thus for application to a movie video the video should be divided into scenes that have a same nature or genre.

CHAPTER # 4

THE PROPOSED ALGORITHM

4.1 PREPROCESSING OF VIDEO

The parent image is Y (luminance) component of video clip frame which is in indexed format. size(176*144).

By taking each frame one by one insert a water marking gray scale image generate a HVS edge detected frame located the watermark insertion pixels in spatial domain.

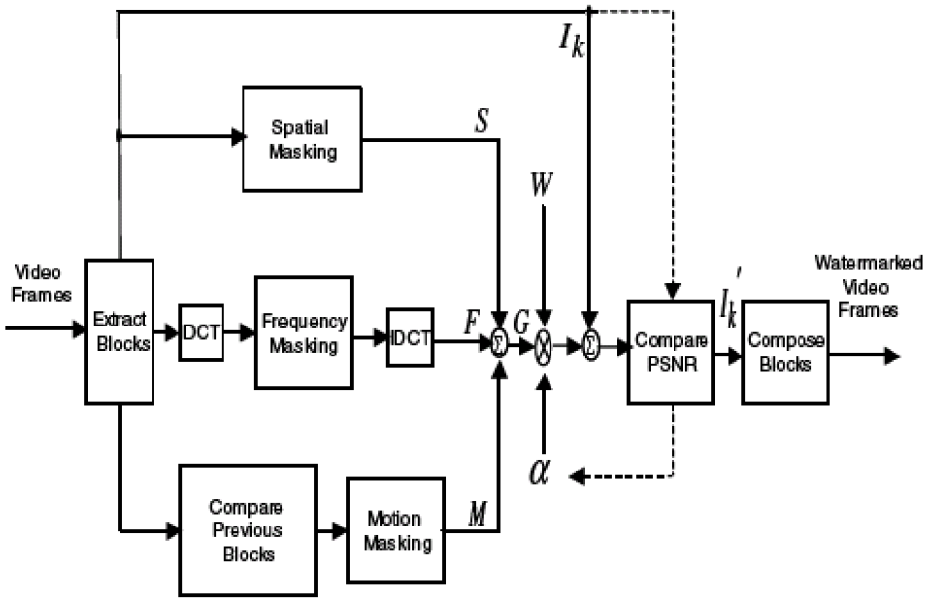


FIG 4.1 PROPOSED ALGORITHM BLOCK DIAGRAM

4.2 WATERMARK INSERTION STEPS

1. Divide the parent image in 4by 4 blocks

$$f(x,y)=\bigcup_{k} f_k(x',y'),\ 0\le x',y'\le 3$$

2.From Global masking map locate the position of pixel in spatial domain which take in to account spatial,motion and frequency masking.

3. Now we embed the linear array data of gray scaled image in the parent image at the particular locations extracted earlier on basis of HVS.

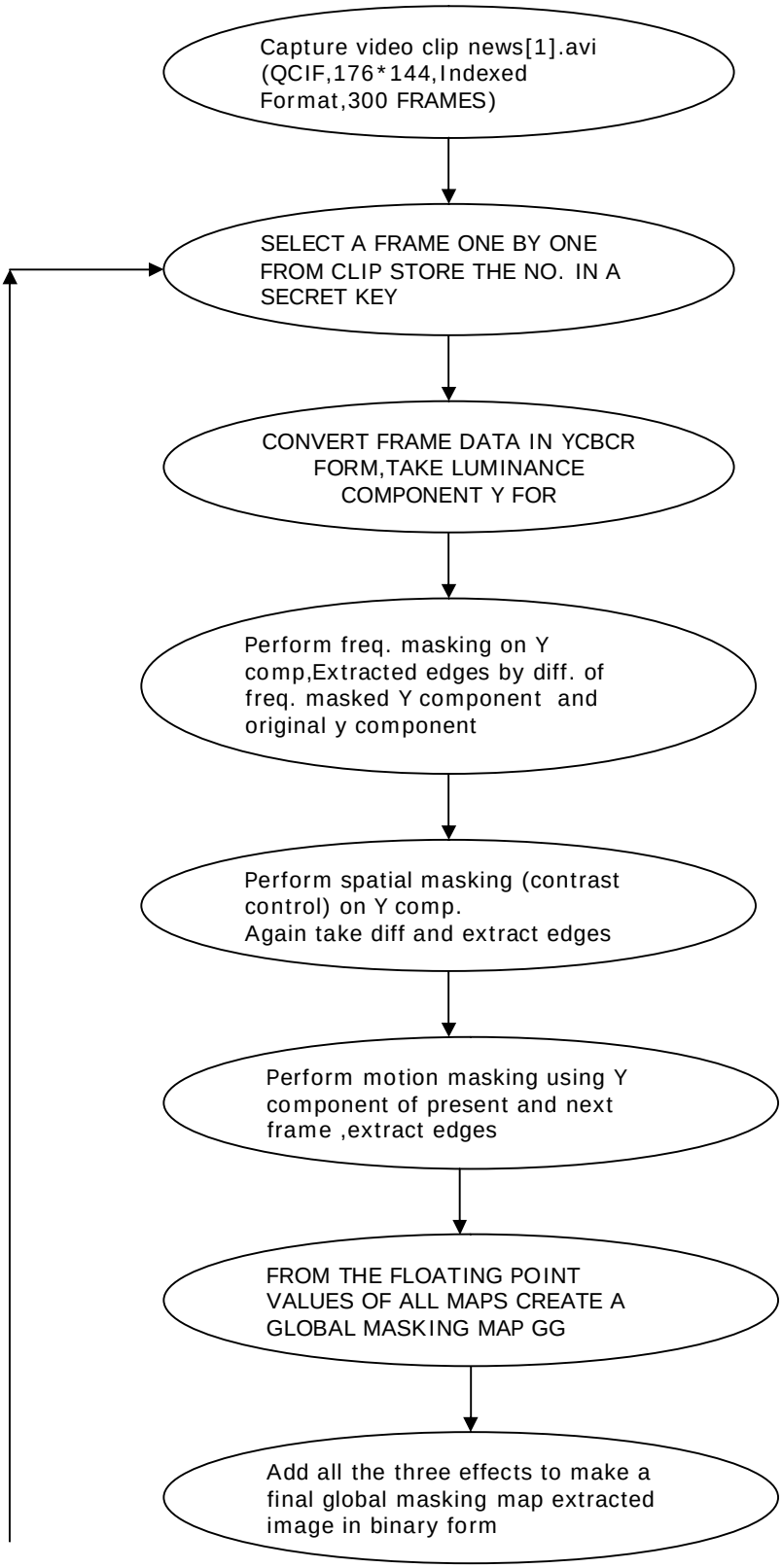
4.3 WATERMARK EXTRACTION STEPS

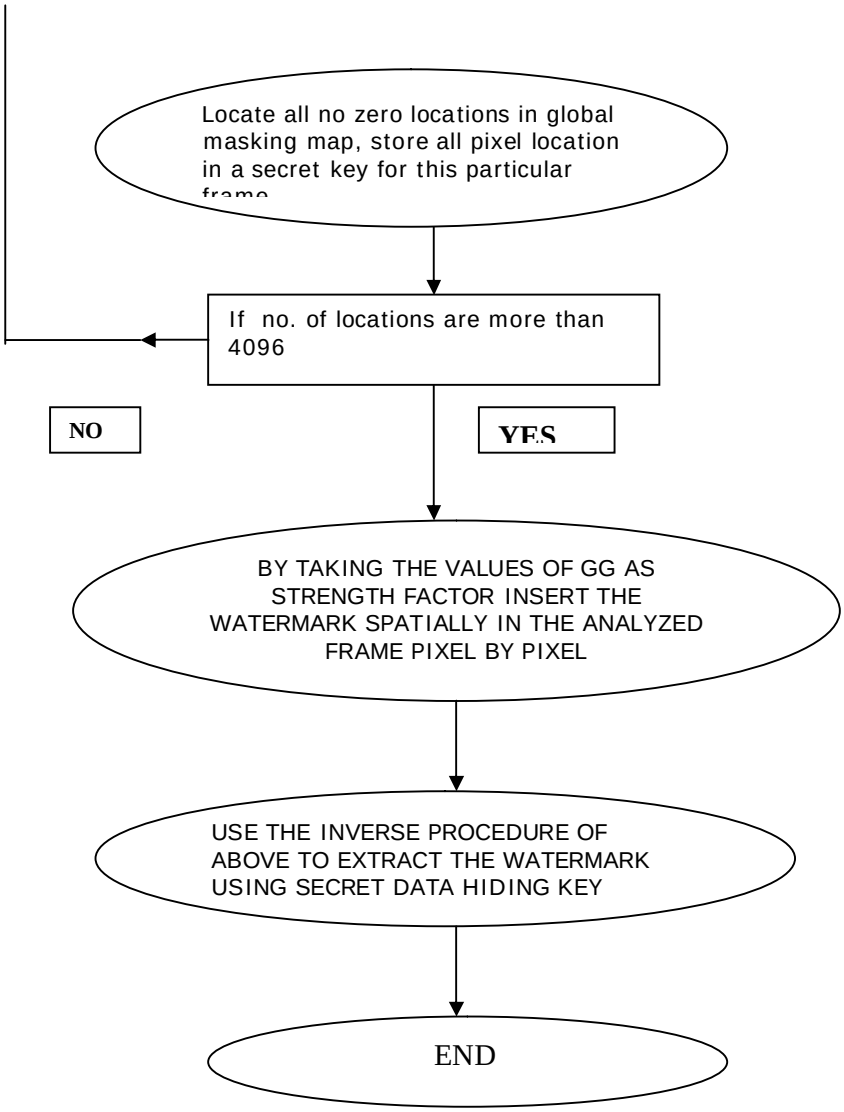
1.The embedded locations are found out by secret key transmitted with the frame .The inverse process of insertion is carried out at those locations.

3. Extracted data is stored in a linear array of size 4096 and then image of size 64*64 is reconstructed .

4.All blocks are settled in the form of an 64*64 image .

4.4 A FLOW CHART OF ALGORITHM STEPS (Fig 4.2)


Capture video clip news[1].avi (QCIF,176*144,Indexed Format,300 FRAMES)
SELECT A FRAME ONE BY ONE FROM CLIP STORE THE NO. IN A SECRET KEY
CONVERT FRAME DATA IN YCBCR FORM,TAKE LUMINANCE COMPONENT Y FOR
Perform freq. masking on Y comp,Extracted edges by diff. of freq. masked Y component and original y component
Perform spatial masking (contrast control) on Y comp. Again take diff and extract edges
Perform motion masking using Y component of present and next frame ,extract edges
FROM THE FLOATING POINT VALUES OF ALL MAPS CREATE A GLOBAL MASKING MAP GG
Add all the three effects to make a final global masking map extracted image in binary form


Locate all no zero locations in global masking map, store all pixel location in a secret key for this particular frame
If no. of locations are more than 4096
NO
YES
BY TAKING THE VALUES OF GG AS STRENGTH FACTOR INSERT THE WATERMARK SPATIALLY IN THE ANALYZED FRAME PIXEL BY PIXEL
USE THE INVERSE PROCEDURE OF ABOVE TO EXTRACT THE WATERMARK USING SECRET DATA HIDING KEY
END

CHAPTER # 5

PICTORIAL VIEW OF SIMULATION

Luminance component of the frame to be watermarked

FIG 5.1

DCT of the luminance component to be watermarked

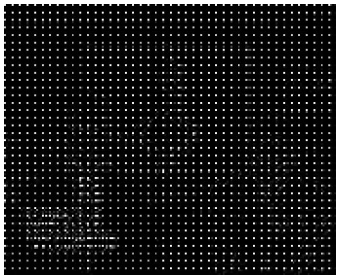


FIG 5.2

DCT Freq. masked component of IY

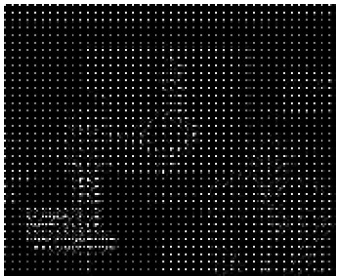


FIG 5.3

Freq. masked image of IY

FIG 5.4

Diff. of IY and freq masked image

FIG 5.5

EDGE remain after freq. masking

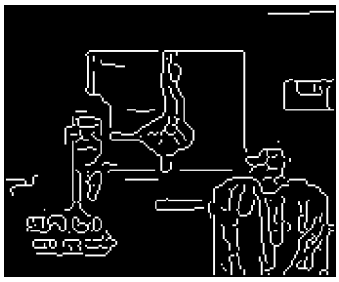


FIG 5.6

CONTRAST controlled image

FIG 5.7

EDGES of contrast controlled image

FIG 5.8

LUMINANCE component of first image

FIG 5.9

LUMINANCE COMPONENT OF NEXT FRAME

FIG 5.10

DIFFERENCE OF TWO choosen and next frame

FIG 5.11

edge of diff. between two frames

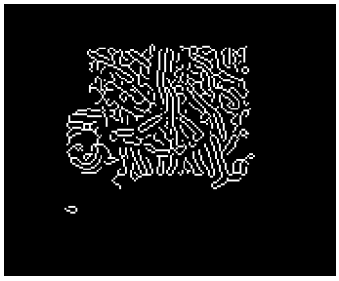


FIG 5.12

FIG 5.13

total edge due to freq,spatial,motion masking

IMAGE TO BE WATERMARKED

FIG 5.14

MAGE AFTER APPLYING TOTAL GLOBAL MASKING MAI

FIG5.15

FIG 5.16

DCT of image to be watermarked

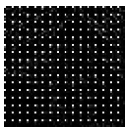


FIG 5.17

WATERMARKED IMAGE Y component

indexed water marked image

FIG 5.18

FIG 5.19
EXTRACTED WATERMARK

CHAPTER # 6

RESULT AND DISCCUSSION

DETAIL OF VIDEO CLIP TO BE INSERTED AS WATERMARK

```
             Filename: 'for.avi'
             FileSize: 1240576
          FileModDate: '05-Sep-2006 13:08:10'
            NumFrames: 300
       FramesPerSecond: 1
                 Width: 64
                Height: 64
            ImageType: 'indexed'
      VideoCompression: 'none'
               Quality: 0
    NumColormapEntries: 256
```

DETAIL OF VIDEO CLIP TO BE USED AS PARENT FOR WATERMARK INSERTION

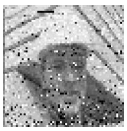
```
             Filename: 'news[1].avi'
             FileSize: 7614976
          FileModDate: '08-Jun-2006 06:17:46'
            NumFrames: 300
       FramesPerSecond: 25
                 Width: 176
                Height: 144
            ImageType: 'indexed'
      VideoCompression: 'none'
               Quality: 0
    NumColormapEntries: 256
```

NO. OF PIXEL LOCATION AVAILABLE IN EACH FRAME AFTER USING HVS BY GLOBAL MASKING

4334 NO. OF PIXEL LOCATIONS FOR FRAME NO.1
5861 NO. OF PIXEL LOCATIONS FOR FRAME NO.3
5830 NO. OF PIXEL LOCATIONS FOR FRAME NO.8
4274 NO. OF PIXEL LOCATIONS FOR FRAME NO.12
5498 NO. OF PIXEL LOCATIONS FOR FRAME NO.13
4695 NO. OF PIXEL LOCATIONS FOR FRAME NO.16
4293 NO. OF PIXEL LOCATIONS FOR FRAME NO.17
5943 NO. OF PIXEL LOCATIONS FOR FRAME NO.18
4148 NO. OF PIXEL LOCATIONS FOR FRAME NO.20
4652 NO. OF PIXEL LOCATIONS FOR FRAME NO.21
4235 NO. OF PIXEL LOCATIONS FOR FRAME NO.22
5887 NO. OF PIXEL LOCATIONS FOR FRAME NO.23
4427 NO. OF PIXEL LOCATIONS FOR FRAME NO.25
5943 NO. OF PIXEL LOCATIONS FOR FRAME NO.28
4709 NO. OF PIXEL LOCATIONS FOR FRAME NO.29
4443 NO. OF PIXEL LOCATIONS FOR FRAME NO.30

FIG TABLE 6.1

JPEG; Quality-100%



SNR of JPEG; Quality-100 NC = 88.2810 FIG 6.1

JPEG; Quality-80%

THE EXTRACTED WATERMARK

SNR of JPEG; Quality-80 NC = 84.2144 FIG 6.2

JPEG; Quality-50%

THE EXTRACTED WATERMARK

SNR of JPEG; Quality-50 NC = 72.3845 FIG 6.3

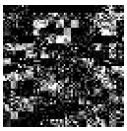
JPEG; Quality-30%

THE EXTRACTED WATERMARK

SNR of JPEG; Quality-30 NC = 52.6063 FIG 6.4

JPEG; Quality-10%

THE EXTRACTED WATERMARK



SNR of JPEG; Quality-10 NC = 24.3253 FIG 6.5

Average Filtered Image

THE EXTRACTED WATERMARK

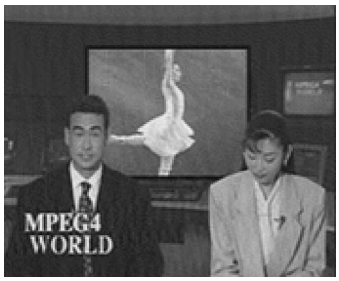
SNR of average filtered image= 13.635275 NC = 7.5416 FIG 6.6

Blurred Image

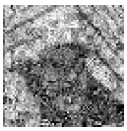
THE EXTRACTED WATERMARK

SNR of blurred image = 12.103189 NC = 9.7227 FIG 6.7

De-Blurred, True PSF



THE EXTRACTED WATERMARK



SNR of de-blurred image = 24.036424 FIG 6.8
NC = 74.5287

Rotated Image

THE EXTRACTED WATERMARK

SNR of rotated image = 6.698496 FIG 6.9 NC = 17.4861

Salt-n-Pepper Noise Affected Image

THE EXTRACTED WATERMARK

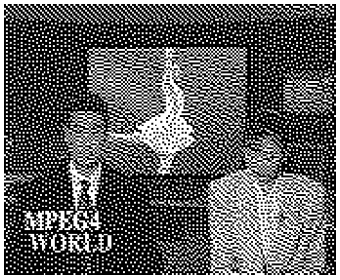
SNR OF SALT N PEPER NOISE IMAGE = 26.737624 FIG 6.10
NC = 86.9501

AWGN Affected Image

THE EXTRACTED WATERMARK

SNR of AWGN noised image= 14.238205 FIG 6.11
NC = 71.0536

Dithered Image



THE EXTRACTED WATERMARK

SNR for dithered image= -1.171755 FIG 6.12
NC = 63.9926

Median Filtered Image

THE EXTRACTED WATERMARK

SNR of median filtered image= 0.024468 FIG 6.13
NC = 17.1904

sharp Filtered Image

THE EXTRACTED WATERMARK

SNR of sharpen filtered image= 4.035836 FIG 6.14
NC = 86.5434

NOTE : ALL ATTACKS ARE PERFORMED ON THE Y COMPONENT OF THE FRAME

MPEG ATTACK

EXTRACTED FRAME 1 NC = 47.5046

EXTRACTED FRAME 13 NC = 42.0333

EXTRACTED FRAME 17 NC = 41.1091

EXTRACTED FRAME 18 NC = 40.5176

VIDEO SCENE CHARACTERSTIC IMPROVEMENT RESULTS

THE MEAN ABSOLUTE LOW DIFFERENCE TABLE-6.1

SEQUENCE	HTAL	LTAL	HTLL	LTLL
Container[2].avi	39.6608	53.7798	**15.6057**	18.7059
Coastguard[2].avi	53.7798	51.440	33.6191	**19.1484**
Akiyo[2].avi	37.0589	53.7798	**20.9115**	53.7798
mother[22].avi	44.9441	53.7798	**49.8873**	53.7798
foreman[2].avi	37.9814	53.7798	22.69960	**10.9120**
news[2].avi	39.1010	53.7798	**26.0757**	53.7798
hall[2].avi	33.6395	53.7798	**31.5998**	53.7798
silent[2].avi	50.0187	53.7798	**25.1424**	38.9213

WATERMARK EMBEDDING CRITERIA

LTAL : Low-texture any luminance(random)

LTLL : Low-texture low luminance

HTAL : High-texture any luminance(random)

HTLL : High-texture low luminance

TABLE-6.2

SEQUENCE	PREFFERED WATERMARK EMBEDDING METHOD
Container[2].avi	**HTLL**
Coastguard[2].avi	**LTLL**
Akiyo[2].avi	**HTLL**
mother[22].avi	**HTLL**
foreman[2].avi	**LTLL**
news[2].avi	**HTLL**
hall[2].avi	**HTLL**
silent[2].avi	**HTLL**

TABLE-6.3 SEQUENCE CHARACTERSTICS

SEQUENCE	LDLTR	LTL	LTC	LDHTR	HTL	HTC
Container[2].avi	0.0014	0.5688	**0.0024**	0.00035	0.5279	0.00067
Coastguard[2].avi	0.3246	0.4711	0.6890	0.3233	0.4662	**0.6937**
Akiyo[2].avi	0.6352	0.3604	**1.7624**	0.4158	0.4255	0.9773
mother[22].avi	0.0440	0.4571	**0.0962**	0.0071	0.4463	0.0159
foreman[2].avi	0.1497	0.6384	0.2345	0.1282	0.5423	**0.2364**
news[2].avi	0.8009	0.2932	**2.7319**	0.5059	0.3885	1.3022
hall[2].avi	0.0105	0.5482	**0.0192**	0.0065	0.5340	0.0121
silent[2].avi	0.3370	0.5040	**0.6688**	0.2513	0.4713	0.5332

LDLTR :MEAN LOW DIFFERENCE BLOCKS TO LOW TEXTURE BLOCKS RATIO
LTL :MEAN LOW TEXTURE LUMINANCE
LTC :LOW TEXTURE CHARACTERSTICS
LDHTR :MEAN LOW DIFFERNCE BLOCKS TO HIGH TEXTURE BLOCKS RATIO
HTL :MEAN HIGH TEXTURE LUMINANCE
HTC :HIGH TEXTURE CHARACTERSTICS

news[2].avi

akiyo[2].avi

Coastguard[2].avi

container[2].avi

Mother[2].avi

silent[2].avi

Foreman[2].avi

hall[2].avi

CHAPTER # 7

7.1 CONCLUSION AND FUTURE WORK

Digital watermarking has recently been ex-extended from still images to video content .Further research in this area is strongly motivated by an increasing need from the copyright owners to reliably protect their rights. Because of the large economic stakes, digital watermarking is promised to a great future. New applications are likely to emerge and may combine existing approaches. For example, a watermark can be separated in to two parts: one for copy right protection and the other for customer fingerprinting . Robust ness has to be considered attentively. There are indeed many non hostile video processing which might alter the watermark signal. It might not even be possible to be immune against all those attacks and detailed constraints has to be defined according to the targeted application. Since collusion is far more critical in the context of video, it must be seriously considered. Finally the real-time constraint has to be met in many applications.Inspite of all those challenges, many algorithms have already been proposed in the literature. It goes from the simple adaptation of a watermarking algorithm for still images to the really video specific watermarking scheme. Open paths still remain in video watermarking. This technology is indeed in its infancy and is far from being as mature as for still images. Quite all possible image processing have been investigated for still images watermarking. On their side, the proposed algorithms for video have remained relatively simple. Many video processing have not been tried and the line is consequently not exhausted.

In this thesis report, we propose a new masking model for video watermarking based on the characteristics of the human visual system (HVS). In order to design the general watermarking scheme, we embed the watermark signals in the uncompressed video sequence. In this proposed algorithm we define an HVS-optimized global masking map for

the best trade-of between invisibility and robustness. We generate the global masking map by combining the frequency, the spatial and the motion masking effects. After embedding the watermark signal using the information from the global masking map, we control the amount of watermarks with the control parameters. Experimental results show that the proposed method is imperceptible to human eyes, and also good in terms of watermark capacity. We have observed that the logo images under those attacks are extracted properly only with slight degradation of image quality.

The proposed algorithm could survive such as additive noise, rescaling, line removal attacks and JPEG compression well. For JPEG compression, the watermark was still recognizable even at the low quantization factor of 30.

Moreover, introduction of perceptual measures have significantly improved the performances of algorithms for still images. This approach has been extended to video by our video scene characterstic detection . Perceptual measures for video exist but the major challenge consists in being able to exploit the real-time. Finally, the second generation of watermarking algorithms has only given its first results. Future discoveries in this domain are likely to be of great .

8. BIBILOGRAPHY

[1] A Video Watermarking Algorithm Based on the Human Visual System Properties Ji-Young Moon and Yo-Sung Ho SamsungElectronicsCo.,Paldal-gu,Suwon-si,Gyenggi-do,Korea Kwangju InstituteofScienceandTechnology(K-JIST) Kwangju, **500-712,Korea**

[2] An Efficient Digital Image-in-Image Watermarking Algorithm Using the Integer Discrete Cosine Transform (IntDCT) . J. Zhang, Anthony T. S. Ho ICICS-PCM 2003 15-18 December 2003 Singapore IEEE 2003

DIVISION OF INFORMATION ENGINEERING SCHOOL OF **ELECTRICAL AND ELECTERONICS ENGINEERING ,NANYANG TECHNOLOGICAL UNIVERSITY ,NANYANG AVENUE ,SINGAPORE 639798**

[3] Video scene characterstic detection to improve digital watermarking transparency I.W.Agung and P.Sweeney IEE Proc. Visual Image Signal Process, Vol .151 ,No.2 April 2004

[4] R.C. Gonzalez and R.E. Woods - Digital Image Processing; Pearson Education -Asia, 1992.

[5] J.G. Proakis and D.G. Manolakis - Digital Signal Processing – Principles, Algorithms and Applications; Third Edition; Prentice Hall of India, 2003.

[6] S.K. Mitra – Digital Signal Processing – A Computer Based Approach; Tata-McGraw Hill.

[7] S.J. Chapman – MATLAB® Programming for Engineers, Second Edition; Thomson 2003.

[8] Lain E.G. Richardson H.264 and MPEG-4 video compression video coding for next generation Multimedia

[9] C. Cachin, "An Information-Theoretic Model for Steganography", Proceedings of 2nd Workshop on Information Hiding, MIT Laboratory for Computer Science, May 1998

[10] R. Popa, An Analysis of Steganographic Techniques, The "Politehnica" University of Timisoara, Faculty of Automatics and Computers, Department of Computer Science and Software

Engineering, http://ad.informatik.uni-freiburg.de/mitarbeiter/will/dlib_bookmarks/digital-watermarking/popa/popa.pdf, 1998

[11] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, "Information Hiding - A Survey", Proceedings of the IEEE, vol. 87, no. 7, pp. 1062-1078, July 1999

[12] M. D. Swanson, B. Zhu and A. H. Tewfik, "Robust Data Hiding for Images", IEEE Digital Signal Processing Workshop, pp. 37-40, Department of Electrical Engineering, University of Minnesota, http://www.assuredigit.com/tech_doc/more/Swanson_dsp96_robust_datahiding.pdf, September 1999

[13]voloshynovskiy Sviatolsav, Joachim J. and Su Jonathan K., "Attacks on Digital Watermarks:Classification, Estimation-Based Attacks, and Benchmarks", Proceedings of IEEE Communications Magazine, vol. 39, pages 118--126, August 2001.

[14]Wolfgang R. B. and Delp E. J., "A Watermark for Digital Images", In Proceedings of IEEE, pages. 219-222, 1996.

[15] C. I. Podilchuk and W. Zeng, "Image-adaptive watermarking using visual models," IEEE J. Select. Areas Commun., vol. 16, pp. 525–539, May 1998.

[16] An efficient watermark embedding technique Ahmed A. Ab delwahab, E.M. Saad ,Nessrin I. Yassin NRSC(2005) March 15-17 ,2005 cairo egypt

[17] IEEE website : www.ieee.org

[18] www.mathsworks.com

APPENDIX - A

```
clc;
clf;
close all;
clear all;

%============READ A MOVIE TAKE A FRAME==================================%

beta = 0.7;
alpha = 0.649;
watermark_frames = 30;

water = aviinfo('for.avi')
watm  = aviread('for.avi',1:30);

asd   = aviinfo('news[1].avi')
mob   = aviread('news[1].avi',1:300);


%-------------------------IMAGE TO BE WATERMARKED ----------------------%

%frame_no = rand(1,100);
%frame_no = frame_no*1000;

%counter = 0;

for frame=1:1:watermark_frames


    %if(frame_no(frame)<300  && counter < watermark_frames)
     %   counter=counter+1;
      %  n = fix(frame_no(frame));
       % key(counter)=n;
        %n1 = n+1;
    %else
     %   continue;
    %end
    n = frame;
    n1= n+1;

aa = mob(n).cdata;
bb = mob(n).colormap;

pp = ind2rgb(aa,bb);
pp = rgb2ycbcr(pp);


IY = pp(:,:,1);


%figure(100+frame);
%imshow(IY);
%title('Luminance component of the frame to be watermarked');

ICr = pp(:,:,2);
```

```
ICb = pp(:,:,3);


%========================== FREQUENCY MASKING TRIAL ====================%

[ROW,COL] = size(IY);
BCK       = (ROW *COL)/(4*4);
k         = zeros(4);
T2        = zeros(BCK,4,4);
Tth       = zeros(BCK,4,4);
T3        = zeros(ROW,COL);
T1        = zeros(ROW,COL);
S3        = zeros(ROW,COL);

%---------------------------4*4 DCT OF IMAGE -----------------------------%

c=0;
k = dctmtx(4);
for ii=1:4:ROW
    for jj=1:4:COL
        x=ii+3;
        y=jj+3;
        c=c+1;
        ss=IY(ii:x,jj:y);
        P=(k * ss * k');
        T2(c,1:4,1:4)=P;
        T1(ii:x,jj:y)=P;
    end
end

%figure(2);
%imshow(T1);
%title('DCT of the luminance component to be watermarked');

%------------------FREQ MASKING START ----------------------------------%
p = 0;

for ii=1:1:BCK
    p = p + T2(ii,1,1);
end

p=p/BCK;


for c=1:1:BCK
    for ii=1:1:4
        for jj=1:1:4
            Tth(c,ii,jj) =  T2(c,ii,jj) *  power((T2(c,1,1)/p),alpha);
        end
    end
end

%--------------STORING OF VARIOUS BLOCKS IN IMAGE ARRAY ------------------%

c=0;

for ii=1:4:ROW
    for jj=1:4:COL
        x=ii+3;
```

```
        y=jj+3;
        c=c+1;
        T3(ii:x,jj:y) = Tth(c,1:4,1:4);
    end
end

%figure(3);
%imshow(T3);
%title('DCT Freq. masked component of IY');
%----------------------------INVERSE DCT IMAGE---------------------------%

for ii=1:4:ROW
    for jj=1:4:COL
        x=ii+3;
        y=jj+3;
        ss = T3(ii:x,jj:y);
        P =  (k' * ss * k);
        S3(ii:x,jj:y)=P;
    end
end
%figure(4);
%imshow(S3);
%title('Freq. masked  image of IY');
%-----------------------------------------------------------------------------%

kk1 = imsubtract(S3,IY);

%figure(5);
%imshow(kk1);
%title('Diff. of IY and freq masked image');


%-----------------EDGE REMAIN AFTER FREQUENCY MASKING--------------------%

edfreq=edge(kk1,'canny');
%figure(6);
%imshow(edfreq);
%title('EDGE remain after freq.  masking');
%======================SPATIAL MASKING TRIAL ============================%

[ROW,COL] = size(IY);

S = zeros(ROW,COL);

a = 0.05;

for ii=1:ROW
    for jj=1:COL
           S(ii,jj) = 1+(99 * ( log10(1+IY(ii,jj)*a)-
log10(1+a))/(log10(1+100*a)-log10(1+a)));
    end
end

%figure(7);
%imshow(S);
%title('CONTRAST controlled image');

edcont = edge (S ,'canny');
```

```
%figure(8);
%imshow(edcont);
%title('EDGES of contrast controlled image');

%============================MOTION MASKING EFFECT=======================%

aa = mob(n).cdata;
bb = mob(n).colormap;



pp=ind2rgb(aa,bb);
pp=rgb2ycbcr(pp);


IYY = pp(:,:,1);

%figure(9);
%imshow(IYY);
%title('LUMINANCE component of first image');



aa1 = mob(n1).cdata;
bb1 = mob(n1).colormap;

pp1=ind2rgb(aa1,bb1);
pp1=rgb2ycbcr(pp1);

IYZ = pp1(:,:,1);

%figure(10);
%imshow(IYZ);
%title('LUMINANCE COMPONENT OF NEXT FRAME');



kk = imsubtract(IYY,IYZ);
kk = im2double(kk);

%figure(11);
%imshow(kk);
%title('DIFFERENCE OF TWO choosen and next frame');

edmotion = edge (kk ,'canny');

%figure(12);
%imshow(edmotion);
%title('edge of diff. between two frames');

%=========================================================================%
S = S;
M = kk;
F = kk1;
GG(frame,:,:)= S+M+F;
%figure(14);
%imshow(GG);
%title('GAURAV');
```

```
S = im2double(edcont);
M = im2double(edmotion);
F = im2double(edfreq);

G(frame,:,:)= S+M+F;

%figure(13);
%imshow(G);
%title('total edge due to freq,spatial,motion masking');



%=========================================================================%

c=0;
for ii=1:1:ROW
    for jj=1:1:COL
        if( G(frame,ii,jj)==1)
           c = c+1;
           pixel_row(frame,c)=ii;
           pixel_col(frame,c)=jj;
        end
    end
end

cdd(frame)=c;

if(cdd(frame) > 4096)

fprintf('%d NO. OF PIXEL LOCATIONS FOR FRAME NO.%d \n',cdd(frame),frame);

aaa = watm(1).cdata ;
bbb = watm(1).colormap;

RGB =  ind2rgb(aaa,bbb);
A   =  rgb2gray(RGB);

W = im2double(A);

%figure(17);
%imshow(W);
%title('IMAGE TO BE WATERMARKED');

[WR,WC]=size(W);

[ROW COL]=size(IY);


c=0;
for row=1:1:WR
    for col=1:1:WC


          c=c+1;

          IY( pixel_row(frame,c),pixel_col(frame,c)) =
IY(pixel_row(frame,c),pixel_col(frame,c))+(0.05 *
GG(frame,pixel_row(frame,c),pixel_col(frame,c)) * W(row,col)) ;
```

```
    end
end

%figure(200+frame);
%imshow(IY);
%title('WATERMARKED IMAGE Y component ');

ppw(:,:,1)  = IY(:,:);
%mobw(frame,:,:) = IY(:,:);
ppw(:,:,2)  = ICr;
ppw(:,:,3)  = ICb ;

%figure(300+frame);
%imshow(ppw);
%title(' WATERMARKED IMAGE ');


ppw1(frame,:,:,:) = ycbcr2rgb(ppw);
mobtemp(:,:,:)=ppw1(frame,:,:,:);
[mob(frame).cdata , mob(frame).colormap] =
rgb2ind(mobtemp,mob(frame).colormap);


figure(400+frame);
imshow(mob(frame).cdata , mob(frame).colormap);
title('indexed water marked image');

     end
end


movie2avi(mob,'newsw.avi',
'colormap',mob(1).colormap,'compression','none','fps',1);
movie2avi(mob,'newsw10.avi','colormap',mob(1).colormap,'compression','none',
'fps',10);
movie2avi(mob,'newsw20.avi','colormap',mob(1).colormap,'compression','none',
'fps',20);

%video_hvs(ppw1,watm,pixel_row,pixel_col,cdd,GG,watermark_frames);

% MPEG attack ======================%
mov= mpgread('NEWS10.m1v', [1:100],'indexed');
for i=1:1:30
[g,map] = frame2im(mov(i));
x = ind2rgb(g,map);
y = rgb2ycbcr(x);
wm =y(:,:,1);
%wm = im2double(wm);
figure;clf;
imshow(wm);
mobtemp(:,:,1)=wm;
mobtemp(:,:,2)=y(:,:,2);
mobtemp(:,:,3)=y(:,:,3);
video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,i)
end
%s(ii) = snr_cal(B1,f1);
%fprintf('SNR = %f\n',s(ii));
```

```
%frame dropping and averaging ================%




ii = 1;

mobtemp(:,:,:) = ppw1(ii,:,:,:) ;
mobtemp(:,:,:) = rgb2ycbcr(mobtemp(:,:,:));
S3(:,:) = mobtemp(:,:,1);
B1=im2double(S3);

imwrite(S3, 'J100%.jpg','quality',100);
ii=1;
J = imread('J100%.jpg');
figure;clf;
imshow(J,[0 255],'truesize');title('JPEG; Quality-100%');
f1=im2double(J);
mobtemp(:,:,1)=f1;
%mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
s(ii) = snr_cal(B1,f1);

fprintf('SNR = %f\n',s(ii));
%fprintf('NC 100= %f\n',NC(ii));



imwrite(S3, 'J80%.jpg','quality',80);
ii=2;
J = imread('J80%.jpg');
figure;clf;
imshow(J,[0 255],'truesize');title('JPEG; Quality-80%');
f1=im2double(J);
mobtemp(:,:,1)=f1;
%mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
s(ii) = snr_cal(B1,f1);
fprintf('SNR = %f\n',s(ii));


imwrite(S3, 'J60%.jpg','quality',60);
ii=3;
J = imread('J60%.jpg');
figure;clf;
imshow(J,[0 255],'truesize');title('JPEG; Quality-60%');
f1=im2double(J);
mobtemp(:,:,1)=f1;
%mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
s(ii) = snr_cal(B1,f1);
fprintf('SNR = %f\n',s(ii));


imwrite(S3, 'J40%.jpg','quality',40);
ii=4;
```

```
J = imread('J40%.jpg');
figure;clf;
imshow(J,[0 255],'truesize');title('JPEG; Quality-40%');
f1=im2double(J);
mobtemp(:,:,1)=f1;
%mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
s(ii) = snr_cal(B1,f1);;
fprintf('SNR = %f\n',s(ii));



imwrite(S3, 'J30%.jpg','quality',30);
ii=5;
J = imread('J30%.jpg');
figure;clf;
imshow(J,[0 255],'truesize');title('JPEG; Quality-30%');
f1=im2double(J);
mobtemp(:,:,1)=f1;
%mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
s(ii) = snr_cal(B1,f1);
fprintf('SNR = %f\n',s(ii));
imwrite(S3, 'J20%.jpg','quality',20);
ii=6;
J = imread('J20%.jpg');
figure;clf;
imshow(J,[0 255],'truesize');title('JPEG; Quality-20%');
f1=im2double(J);
mobtemp(:,:,1)=f1;
%mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
s(ii) = snr_cal(B1,f1);
fprintf('SNR = %f\n',s(ii));

%Blurring - De-blurring Attack:

           ii=7;

           % create PSF
           LEN = 2;
           THETA = 1;
           PSF = fspecial('motion',LEN,THETA);

           % Blur the image
           Blurred = imfilter(S3,PSF,'circular','conv');
           imwrite(Blurred,'BLURR.tif');
           figure;clf;
           imshow(Blurred,'truesize');title('Blurred Image');


           J=imread('BLURR.tif');
          f1=im2double(J);
          mobtemp(:,:,1)=f1;
 %         mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
          video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
          s(ii) = snr_cal(B1,f1);
          fprintf('SNR = %f\n',s(ii));
          ii=ii+1;
```

```
          % Deblur the image
          h10 = deconvwnr(Blurred,PSF);
          imwrite(h10, 'DE_BLURRED.tif');

          figure; clf;
          imshow(h10,'truesize');title('De-Blurred, True PSF');

          J=imread('DE_BLURRED.tif');
         f1=im2double(J);
         mobtemp(:,:,1)=f1;
 %        mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
         video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
         s(ii) = snr_cal(B1,f1);
          fprintf('SNR = %f\n',s(ii));
         ii=ii+1;

           %Rotate image by 4 degrees
           WM_Ir = imrotate(S3,-2,'bilinear');
           WM_Ir2 = imresize(WM_Ir,[144 176]);
           imwrite(WM_Ir2,'rot4.tif');

           figure;clf;
           imshow(WM_Ir2,[],'truesize') ;title('Rotated Image');

           J=imread('rot4.tif');
           f1=im2double(J);
           mobtemp(:,:,1)=f1;
  %         mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
           video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
            s(ii) = snr_cal(B1,f1);
            fprintf('SNR = %f\n',s(ii));
            ii=ii+1;
           %Rotate back by 5 degrees and crop irrelevent portion
            WM_Irr = imrotate(WM_Ir,+4,'bilinear');
            rs_11 = imresize(WM_Irr,[144 176]);
            h_11 = imcrop(rs_11,[80 80 500 500]);
            h11 = imresize(h_11,[144 176]);
            imwrite(h11,'rot_back.tif');
            figure;clf;
            imshow(h11,'truesize');title('Rotation Corrected cropped and
resized Image');
             J=imread('rot_back.tif');
             f1=im2double(J);
             mobtemp(:,:,1)=f1;
   %          mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
             video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
             s(ii) = snr_cal(B1,f1);
             fprintf('SNR = %f\n',s(ii));
             ii=ii+1;

             %Average Filtering

             F_h = ones(3,3)/9;
             h12 = imfilter(S3,F_h);
             imwrite(h12, 'Fltr_avg.tif');

             figure;clf;
             imshow(h12,'truesize');title('Average Filtered Image');
```

```
            J=imread('Fltr_avg.tif');
            f1=im2double(J);
            mobtemp(:,:,1)=f1;
     %       mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
            video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
            s(ii) = snr_cal(B1,f1);
            fprintf('SNR = %f\n',s(ii));
            ii=ii+1;
            %sharpening  Filtering

            F_h = [-1,-1,-1;-1,8,-1;-1,-1,-1]/1;

            h12 = imfilter(S3,F_h);
            figure;
            imshow(h12);
            h12 = imadd(S3,h12);
            imwrite(h12, 'Fltr_sharp.tif');

            figure;clf;
            imshow(h12,'truesize');title('sharp Filtered Image');

            J=imread('Fltr_sharp.tif');
           f1=im2double(J);
            mobtemp(:,:,1)=f1;
      %      mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
            video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
            s(ii) = snr_cal(B1,f1);
            fprintf('SNR = %f\n',s(ii));


            ii=ii+1;

            h13 = imnoise(S3,'salt & pepper', 0.002);
            imwrite(h13,'Noisy_SNP.tif');
            figure;clf;
            imshow(h13,'truesize');title('Salt-n-Pepper Noise Affected
Image');

            J=imread('Noisy_SNP.tif');
           f1=im2double(J);
           mobtemp(:,:,1)=f1;
       %    mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
           video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
           s(ii) = snr_cal(B1,f1);
           fprintf('SNR = %f\n',s(ii));
            ii=ii+1;

          % AWG-Noise attack

          h14 = imnoise(S3,'gaussian',.001,.005);
          imwrite(h14,'Noisy_AWGN.tif');
          figure;clf;
          imshow(h14,'truesize');title('AWGN Affected Image');
          J=imread('Noisy_AWGN.tif');
          f1=im2double(J);
          mobtemp(:,:,1)=f1;
        %    mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
          video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
```

```
            s(ii) = snr_cal(B1,f1);
            fprintf('SNR = %f\n',s(ii));
             ii=ii+1;

             % Apply Dithering Attack
             h15 = dither(S3);
             imwrite(h15,  'Im_dithered.tif');
             figure;clf;
             imshow(h15, 'truesize');title('Dithered Image');
             J=imread('Im_dithered.tif');
             f1=im2double(J);
             mobtemp(:,:,1)=f1;
        %    mobtemp(:,:,:)=ycbcr2rgb(mobtemp(:,:,:));
             video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,ii);
             s(ii) = snr_cal(B1,f1);
             fprintf('SNR = %f\n',s(ii));
```

```
MPEG ATTACK
clear all;
close all;
clc;
clf;
water = aviinfo('for.avi')
watm  = aviread('for.avi',1:30);

mov= mpgread('NEWS10.m1v', [1:100],'indexed');
for i=1:1:30
[g,map] = frame2im(mov(i));
x = ind2rgb(g,map);
y = rgb2ycbcr(x);
wm =y(:,:,1);
%wm = im2double(wm);
figure;clf;
imshow(wm);
mobtemp(:,:,1)=wm;
mobtemp(:,:,2)=y(:,:,2);
mobtemp(:,:,3)=y(:,:,3);
video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,i)
end
%movie2avi(mov,'foren.avi');
%mm = aviread('foren.avi',1:300);
```

```
            function [S] = snr_cal (B1_1,S3_1)
%----------SNR CALCULATION FOR WATERMARKED IMAGE -------------------%


if (size(B1_1) ~= size(S3_1))
   error('The size of image is too large')
end

[m n] = size(B1_1);
```

```
total_I = 0;
total_Dif = 0;

for u = 1:m
     for v = 1:n
         total_I = total_I + B1_1(u, v)^2;
         total_Dif = total_Dif + (B1_1(u, v) - S3_1(u, v))^2;
     end
end
if (total_Dif == 0)
    total_Dif = 1;
end
S = (total_I) / (total_Dif);
S = (10 * log10(S));



function [] =video_hvs1(mobtemp,watm,pixel_row,pixel_col,cdd,GG,i)
%--------------EXTRACTION PROCESS BY INVERSE DCT-------------------------%
mob_orig   = aviread('news[1].avi',1:300);

aaa = watm(1).cdata;
bbb = watm(1).colormap;

RGB = ind2rgb(aaa,bbb);

A   = rgb2gray(RGB);

[WR,WC] = size(A);



orig_W  = im2bw(A);


frame=i;


if(cdd(frame) > 4096)

%aa = mob(frame).cdata;
%bb = mob(frame).colormap;
%mobw(:,:,:)=ppw1(frame,:,:,:);
%pp=ind2rgb(aa,bb);
%mobw(:,:,:) = rgb2ycbcr(mobw(:,:,:));
%mobtemp(:,:,:) = rgb2ycbcr(mobtemp(:,:,:));
HH (:,:) = mobtemp(:,:,1);

aa1 = mob_orig(frame).cdata;
bb1 = mob_orig(frame).colormap;



pp1=ind2rgb(aa1,bb1);
pp1=rgb2ycbcr(pp1);


IY = pp1(:,:,1);
```

```
%figure(17);
%imshow(W);
%title('IMAGE TO BE WATERMARKED');



[ROW COL]=size(IY);


c=0;
for row=1:1:WR
    for col=1:1:WC

     c=c+1;
     W_extract(row,col) = ( HH( pixel_row(frame,c),pixel_col(frame,c)) -
IY(pixel_row(frame,c),pixel_col(frame,c)) ) / ( 0.05  *  GG( frame ,
pixel_row(frame,c) , pixel_col(frame,c)) );

    end
end

figure(frame+500);
imshow(W_extract);

W_4 = im2bw(W_extract);



%--------------NORMALIZED CROSS CORRELATION -------------%

w=0;
r=0;

for ii=1:1:WR
    for jj = 1:1:WC
         w = w + (orig_W(ii,jj) * W_4(ii,jj));
         r = r + (orig_W(ii,jj) * orig_W(ii,jj));
    end
end

fprintf('EXTRACTED FRAME %d',frame);
NC = (w / r)*100



end

TEXTURE1

clc;
clf;
close all;
clear all;



Threshold =3;
w_frames = 30 ;
```

```
DIFF =4;

asd   = aviinfo('mother[22].avi')
mob   = aviread('mother[22].avi',1:300);


temp1=0;
temp2=0;
temp3=0;
temp4=0;

temp5 = 0;
temp6 = 0;
temp7 = 0;
temp8 = 0;

for frame = 1:1:w_frames

n = frame;

aa = mob(n).cdata;
bb = mob(n).colormap;

pp = ind2rgb(aa,bb);
pp = rgb2ycbcr(pp);


IY = pp(:,:,1);

aa1 = mob(n+1).cdata;
bb1 = mob(n+1).colormap;

pp1 = ind2rgb(aa,bb);
pp1 = rgb2ycbcr(pp1);
IY1 = pp1(:,:,1);


[ROW,COL]=size(IY);



k=dctmtx(4);

c=0;
for ii=1:4:ROW
    for jj=1:4:COL

        x=ii+3;
        y=jj+3;
        c=c+1;

         s1=IY1(ii:x,jj:y);
         next4(c,1:4,1:4)= s1;


         s=IY(ii:x,jj:y);
         T4(c,1:4,1:4)= s;
         P=(k * s * k');
```

```
         T2(c,1:4,1:4)=P;
         T1(ii:x,jj:y)=P;
    end
end


%figure(1);
%imshow(T1);



edgemap = edge(IY,'canny');

c =0;
for ii=1:4:ROW
    for jj=1:4:COL

        x=ii+3;
        y=jj+3;
        c=c+1;

         s1 = edgemap(ii:x,jj:y);

         T3(c,1:4,1:4) = s1;

    end
end

blocks = c;
high_texture = 0;
low_texture  = 0;

for c=1:1:blocks
    h=0;
    for ii=1:1:4
        for jj=1:1:4
            if(T3(c,ii,jj)==1)
               h = h+1;
            end
        end
    end
    if(h>=2)
        block_status(frame,c)=1;
        high_texture = high_texture + 1 ;
    else
        block_status(frame,c)=0;
        low_texture = low_texture + 1  ;
    end
end


    high_tex(frame) = high_texture
    low_tex(frame) = low_texture
```

```
RGB=imread('skoda.jpg');
J=rgb2gray(RGB);
J =imresize(J,[50,50],'nearest');
J = im2double(J);


%imshow(J);


%======================== WATERMARKING VARIANTS ======================%

for x=1:1:5

    switch (x)

        case 1

           for c = 1 : 1 : blocks
               WT1(c,1:4,1:4) = T2(c,1:4,1:4);
           end

           i=0;
           j=0;


           tot_lum = 0;
           count = 0;

           for c = 1 : 1 : blocks
               if( block_status(frame ,c)==1)
                   lum = 0;
                   for ii=1:4
                       for jj=1:4
                           lum = lum + T4(c,ii,jj)  ;
                       end
                   end
                   tot_lum = tot_lum + lum;
                   count =  count + 1 ;
                   blk(count)=c   ;



               end

           end
                     ii=0;
                     for i=1:1:50
                           for j=1:1:50
                                ii=ii+1;
                                if(ii<=count)
                                WT1(blk(ii),1,1) = WT1(blk(ii),1,1) +  0.02*
(J(i,j)) ;
                                end
                           end
                     end
```

```
           HTL(frame) = tot_lum / (count * 16);

C=0;
for ii=1:4:ROW
    for jj=1:4:COL
        x=ii+3;
        y=jj+3;
        C=C+1;
        P1=WT1(C,1:4,1:4);
        WWT1(ii:x,jj:y)=P1;
    end
end

%figure;
%imshow(WWT1);
%title('EMBEDDED WATERMARK STRENGTH DCT');

%WRT1 = zeros(frame,1:144,1:176);


c=0;

for ii=1:4:ROW
    for jj=1:4:COL

        x=ii+3;
        y=jj+3;
        c=c+1;

         s = WWT1(ii:x,jj:y);
         P = (k' * s * k);

         %T2(c,1:4,1:4)=P;
         WWT1(ii:x,jj:y)= P;
         %WRT1(frame,ii:x,jj:y)=P;
    end
end


figure;
imshow(WWT1);

c=0;
diff_hightex(frame) =0;
for ii=1:1:ROW
    for jj=1:1:COL

        diff_hightex(frame) = diff_hightex(frame) + (WWT1(ii ,jj) -
IY(ii,jj) );
    end
end


xxx1 = diff_hightex(frame);


%figure;
%imshow(WRT1(frame));
```

```
        case 2


           for c = 1 : 1 : blocks
               WT2(c,1:4,1:4) = T2(c,1:4,1:4) ;
           end

           i=0;
           j=0;


           tot_lum = 0;
           count = 0;


              for c = 1 : 1 : blocks
               if( block_status( frame ,c)==0)


                    lum = 0;

                    for ii=1:4
                        for jj=1:4
                            lum = lum + T4(c,ii,jj)  ;
                        end
                    end


                    tot_lum = tot_lum + lum;
                    count  =  count + 1 ;
                    blk(count)=c;


               end

              end
                    ii=0;
                     for i=1:1:50
                           for j=1:1:50
                                ii=ii+1;
                                if(ii<=count)
                                WT2(blk(ii),1,1) = WT2(blk(ii),1,1) + 0.02 *
(J(i,j)) ;
                                end
                           end
                     end

               LTL(frame) = tot_lum / (count*16);



C=0;
for ii=1:4:ROW
    for jj=1:4:COL
        x=ii+3;
        y=jj+3;
        C=C+1;
```

```
        P1=WT2(C,1:4,1:4);
        WWT2(ii:x,jj:y)=P1;
    end
end

%figure;
%imshow(WWT2);
%title('EMBEDDED WATERMARK STRENGTH DCT');

%WRT2 = zeros(frame,1:144,1:176);


c=0;

for ii=1:4:ROW
    for jj=1:4:COL

        x=ii+3;
        y=jj+3;
        c=c+1;

         s = WWT2(ii:x,jj:y);
         P = (k' * s * k);

         %T2(c,1:4,1:4)=P;
         WWT2(ii:x,jj:y)= P;
        % WRT2(frame,ii:x,jj:y)=P;
    end
end


%figure;
%imshow(WWT2);

c=0;
diff_lowtex(frame) =0;
for ii=1:1:ROW
    for jj=1:1:COL
        diff_lowtex(frame) = diff_lowtex(frame) + (WWT2(ii ,jj) - IY(ii,jj)
);
    end
end


xxx2 = diff_lowtex(frame);


     case 3

             for c = 1 : 1 : blocks
                 WT3(c,1:4,1:4) = T2(c,1:4,1:4) ;
             end

             for c = 1 :2 : blocks

                   count  =  count + 1 ;
                   blk(count)=c;
```

```
            end

            ii=0;
                    for i=1:1:50
                          for j=1:1:50
                               ii=ii+1;
                               if(ii<=count)
                               WT3(blk(ii),1,1) = WT3(blk(ii),1,1) +  0.02*
(J(i,j)) ;
                               end
                          end
                    end


      case 4
                low_tex_low_lum = 0;
                DL =0;
                for c = 1 : 1 : blocks
                 WT4(c,1:4,1:4) = T2(c,1:4,1:4) ;
                end

                i=0;j=0;
           for c = 1 : 1 : blocks
               if( block_status( frame ,c)==0)
                   lum = 0;

                  for ii=1:4
                      for jj=1:4
                          lum = lum + WT4(c,ii,jj)  ;
                      end
                  end

                 if(lum < Threshold )
                     low_tex_low_lum = low_tex_low_lum + 1;
                     blk4(low_tex_low_lum) = c;

                     residue = 0;

                     for kk = 1:4
                          for ll = 1:4
                              residue = residue + abs(T2(c,kk,ll)-
next4(c,kk,ll));
                          end
                     end
                     if(residue < DIFF)
                         DL = DL  + 1 ;
                     end


                      % WT4(c,1,1) = WT4(c,1,1) + J(i,j);

                end

            end

           end

                ii=0;
                    for i=1:1:50
```

```
                            for j=1:1:50
                                 ii=ii+1;
                                 if(ii<=low_tex_low_lum)
                                 WT4(blk4(ii),1,1) = WT4(blk4(ii),1,1) +
0.02* (J(i,j)) ;
                                 end
                            end
                       end

                  DL = DL;
                  LDLTR(frame) = DL/low_tex(frame);
                  low_tex_low_lum = low_tex_low_lum

C=0;
for ii=1:4:ROW
    for jj=1:4:COL
        x=ii+3;
        y=jj+3;
        C=C+1;
        P1=WT4(C,1:4,1:4);
        WWT4(ii:x,jj:y)=P1;
    end
end

%figure;
%imshow(WWT4);
%title('EMBEDDED WATERMARK STRENGTH DCT');

%WRT1 = zeros(frame,1:144,1:176);


c=0;

for ii=1:4:ROW
    for jj=1:4:COL

        x=ii+3;
        y=jj+3;
        c=c+1;

         s = WWT4(ii:x,jj:y);
         P = (k' * s * k);

         %T2(c,1:4,1:4)=P;
         WWT4(ii:x,jj:y)= P;
         %WRT1(frame,ii:x,jj:y)=P;
    end
end


%figure;
%imshow(WWT4);

c=0;
diff_low_tex_low_lum(frame) =0;
for ii=1:1:ROW
    for jj=1:1:COL
```

```
        diff_low_tex_low_lum(frame) = diff_low_tex_low_lum(frame) + (WWT4(ii
,jj) - IY(ii,jj) );
    end
end


xxx4 = diff_low_tex_low_lum(frame);


%figure;
%imshow(WRT1(frame));

       case 5

             high_tex_low_lum = 0;
             for c = 1 : 1 : blocks
                 WT5(c,1:4,1:4) = T2(c,1:4,1:4) ;
             end

                i=0;j=0;
                DH = 0;
              for c = 1 : 1 : blocks
               if( block_status(frame ,c)==1)
                   lum = 0;

                   for ii=1:4
                       for jj=1:4
                           lum = lum + WT5(c,ii,jj)  ;
                       end
                   end
                   if(lum < Threshold )
                       high_tex_low_lum = high_tex_low_lum + 1;
                       blk5(high_tex_low_lum) = c;
                       residue = 0;

                     for kk = 1:4
                          for ll = 1:4
                              residue = residue + abs(T2(c,kk,ll)-
next4(c,kk,ll));
                          end
                     end
                     if(residue < DIFF)
                         DH = DH  + 1 ;
                     end
                       %WT5(c,1,1) = WT5(c,1,1) + J(i,j);

                   end

               end
              end
                    ii=0;
                     for i=1:1:50
                           for j=1:1:50
                                ii=ii+1;
                                if(ii<=high_tex_low_lum)
                                WT5(blk5(ii),1,1) = WT5(blk5(ii),1,1) +
0.02* (J(i,j)) ;
                                end
                           end
```

```
                    end

                DH = DH;
                LDHTR(frame)= DH / high_tex(frame);
                high_tex_low_lum = high_tex_low_lum

C=0;
for ii=1:4:ROW
    for jj=1:4:COL
        x=ii+3;
        y=jj+3;
        C=C+1;
        P1=WT5(C,1:4,1:4);
        WWT5(ii:x,jj:y)=P1;
    end
end

%figure;
%imshow(WWT5);
%title('EMBEDDED WATERMARK STRENGTH DCT');

%WRT1 = zeros(frame,1:144,1:176);


c=0;

for ii=1:4:ROW
    for jj=1:4:COL

        x=ii+3;
        y=jj+3;
        c=c+1;

         s = WWT5(ii:x,jj:y);
         P = (k' * s * k);

         %T2(c,1:4,1:4)=P;
         WWT5(ii:x,jj:y)= P;
         %WRT1(frame,ii:x,jj:y)=P;
    end
end


%figure;
%imshow(WWT5);

c=0;
diff_high_tex_low_lum(frame) =0;
for ii=1:1:ROW
    for jj=1:1:COL

        diff_high_tex_low_lum(frame) = diff_high_tex_low_lum(frame) +
(WWT5(ii,jj) - IY(ii,jj) );
    end
end


xxx5 = diff_high_tex_low_lum(frame);
```

```
    end
end


temp1 = temp1 + LDHTR(frame);
temp2 = temp2 + LDLTR(frame);
temp3 = temp3 + LTL(frame);
temp4 = temp4 + HTL(frame);
temp5 =  temp5 + diff_lowtex(frame);
temp6 =  temp6 + diff_hightex(frame);
temp7 =  temp7 + diff_low_tex_low_lum(frame);
temp8 =  temp8 + diff_high_tex_low_lum(frame);



end

LDHTR_F = temp1/w_frames
LDLTR_F = temp2/w_frames
LTL_F   = temp3/w_frames
HTL_F   = temp4/w_frames

LTC_F = LDLTR_F/LTL_F
HTC_F = LDHTR_F/HTL_F

mean_diff_lowtex  = temp5/w_frames
mean_diff_hightex = temp6/w_frames
mean_diff_low_tex_low_lum = temp7/w_frames
mean_diff_high_tex_low_lum = temp8/w_frames
```