Information Hiding Using Fractal Encoding

Dissertation

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

by

Kamal Gulati
(Roll No: 01329011)

under the guidance of

Prof. Vikram M. Gadre

School of Information Technology
Indian Institute of Technology Bombay
Mumbai

January 2003

To my family.

Abstract

Cryptographic methods are not sufficient to solve all data security related issues. Steganography is the art of hiding information in ways that prevent the detection of hidden messages. Multimedia documents are very easy to copy and distribute in an illicit manner. Copyright labeling is a process that may help to reduce their illicit copying. If this document is copied the copy will also contain the label. This label (or watermark) should be robust enough to withstand normal image processing activities (like image compression, transforming to different format) that do not significantly alter the image appearance.

This thesis investigates the following relevant concepts and terminology. Information hiding and its applications, fractals and image compression using fractal encoding with the main focus being on hiding in the spatial domain. Three information hiding methods are proposed, which are based on the fractal encoding technique, are tested and the results are analyzed.

Acknowledgements

I take this opportunity to express my sincere gratitude towards my guide **Prof. Vikram M. Gadre** for his constant advice, encouragement and cooperation. The regular meeting and discussion therein were invaluable in the realization of this project work. I am extremely grateful to him for his guidance and timely support for preparing this thesis.

Kamal Gulati

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

It is to be expected that digital photographs, videos, and sound tracks will gradually replace their analog counterparts in the near future. Digital representation of signals brings many advantages when compared to analog representations, such as lossless recording and copying, convenient distribution over networks, easy editing and modification, and durable, cheaper, easily reachable archival. Unfortunately, these advantages also present serious problems including wide spread copyright violation, illegal copying and distribution, problematic authentication, and easy forging. Piracy of digital photographs is already a common phenomenon on the Internet. Today, digital photographs or videos cannot be used in the chain of custody as evidence in the court because of nonexistence of a reliable mechanism for authenticating digital images or tamper detection. Information hiding in digital documents provides a means for overcoming those problems.

Depending on what information in which form is hidden in the image, one can distinguish at least two types of data hiding schemes: non-robust, undetectable data hiding, and robust image watermarking. In the first case, a digital image serves as a container for a secret message. Suppose, a spy in a foreign country wants to send messages abroad. He needs to use local communication channels in order to send the messages. He should assume that the communication channel is monitored. Sending encrypted messages would raise suspicion and could result in cutting the access to the communication infrastructure. It is therefore in his best interest to hide the presence of communication at all. This could be solved using a clever steganographic protocol.

In the second application, robust image watermarking, a short message (a watermark) is embedded in the image in a robust manner. By robustness we mean the ability to survive common image processing operations, such as lossy compression, filtering, noise adding, geometrical transformations, etc. Such robust watermark can be obviously used for copyright protection, fraud detection (verification of image integrity), authentication, etc. At this point we emphasize that cryptographic authentication protocols cannot solve all the issues related to authentication. Cryptographic authentication deals with authenticating the sender of the message over insecure channels. However, once the message (image) is decrypted, the image is unprotected and can be copied and further distributed. Unlike classical paintings that can be studied for authenticity using sophisticated experimental techniques, a digital artwork is just a collection of bits. A visible signature in the corner of the image can be easily replaced or removed with advanced image processing software packages, such as PhotoShop. Additional information in the image header can be erased or changed as well. In other words, any attempt to authenticate the digital image by appending information will fail. Digital watermarking provides an appealing alternative by embedding rather than appending information directly into the image itself. The embedded information will be transparent to the human eye, but it should be detectable using a sophisticated algorithm provided a secret key is available.

1.2 Aim

The aim is to develop a new fractal encoding technique, which can find out the possibility to hide maximum amount of data in an image without degrading its quality. Second issue is to make the hidden data robust enough to withstand image processing which do not change the appearance of image. So this technique can also be used for digital watermarking. And also, this technique should be computationally less intensive.

1.3 Organization of Thesis

In **chapter 2**, we discuss steganography and watermarking, their requirements, application, existing techniques and attacks.

In **chapter 3**, we discuss fractals and their application in image compression.

In **chapter 4**, we discuss the fractal techniques to hide and decode the data in image.

In **chapter 5**, we compare these fractal techniques based on results obtained.

Chapter 6 ends with the conclusion and direction for future work.

Chapter 2

Information Hiding

Cryptography hides contents of the communication, but information hiding hides even the presence of the communication. Information hiding in digital image is of two types: Steganography and watermarking.

2.1 Covert Communication (Steganography)

2.1.1 Introduction

The word 'steganography' comes from the Greek steganos (covered or secret) and -graphy (writing or drawing) and thus means, literally, covered writing. It is about exploiting the limited powers of the human visual system. Within reason, any plain text, cipher-text, other images, or anything that can be embedded in a bit stream can be hidden in an image. It does not have to be robust, data should be just invisible.

In contrast to cryptography, where the "enemy" is allowed to detect, intercept and modify messages without being able to violate certain security premises guaranteed by a cryptosystem, the goal of steganography is to hide messages inside other "harmless" messages in a way that does not allow any "enemy" to even detect that there is a second secret message present. Steganography is in the (especially military) literature also referred to as transmission security or short TRANSEC.

2.1.2 Requirements

The most important requirement is that the presence of the hidden message be undetectable. This means that images with and without secret messages should appear identical to all possible statistical tests that can be carried out. It is im-

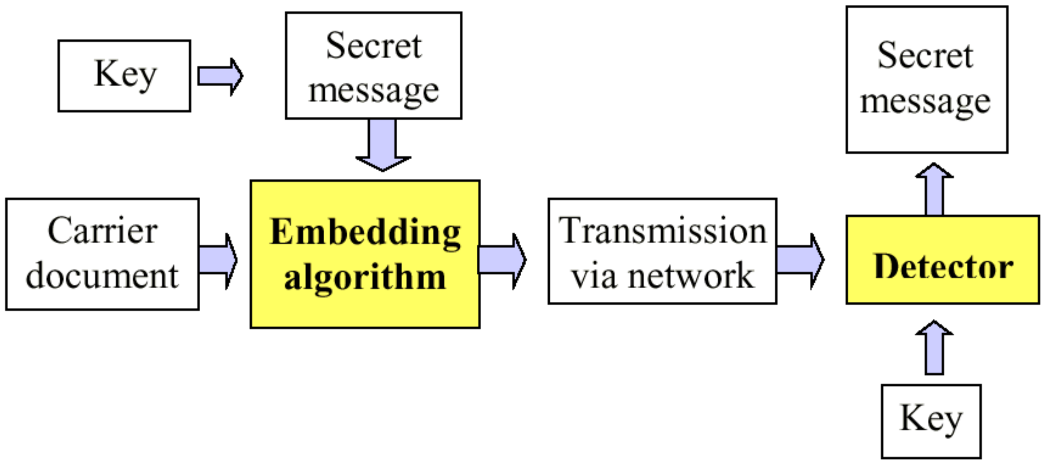


Figure 2.1: Covert communication.

portant to know as much about the statistical properties of the source from which cover images are being drawn as possible. For example, if the images are scanned photographs, there will be stronger correlation in the horizontal direction than in the perpendicular direction. The details of the noise may be specific for each scanner and need to be taken into account if a reliable and secure steganographic protocol is needed. On the other hand, if the images are taken using a digital CCD (Charge Coupled Device) camera, the noise will again have certain specific properties induced by the CCD element and the specific data readout. In either case, the data hiding scheme must respect all known statistical properties of the image source and produce images that cannot be distinguished from images that do not contain any messages.

Another important requirement is the capacity of the communication channel. The challenge is to embed as much information as possible while staying compatible with the image noise model. The last important requirement is that it must be possible to detect the hidden message without the original image.

2.2 Digital Watermarking

2.2.1 Introduction

Digital watermarking is a method of embedding some information which helps for copyright, to authenticate data, identify illegal copies and detect illegal changes made in the data. The purpose of digital watermarks is to provide copyright protection for intellectual property that's in digital format.

Unlike printed watermarks, which are intended to be somewhat visible, digital watermarks are designed to be completely invisible, or in the case of audio clips, inaudible. Moreover, the actual bits representing the watermark must be scattered throughout the file in such a way that they cannot be identified and manipulated. And finally, the digital watermark must be robust enough so that it can withstand normal changes to the file, such as reductions from lossy compression algorithms.

Watermarking schemes that need the original image for watermark extraction are called *non-oblivious*. Typically, such schemes are more robust than *oblivious* schemes that do not need the original image for watermark extraction. On the other hand, the application of non-oblivious schemes is severely limited by the requirement of having the original image available.

2.2.2 Applications and requirements

Watermarking has wide range of application. They can be used for copyright protection of digital images, fingerprinting, adding additional information to digital documents.

Copyright protection of digital images (authentication)

The author of a digital image wants to “sign” the image so that no one else can claim the authorship of the image to himself. The signature cannot be appended to the image file, nor can it be visibly imprinted on the image because such signatures can be easily removed or replaced. Cryptographic digital signatures cannot be applied because images are to be viewed by others and, therefore, will be distributed “in plain”. Cryptographic digital signatures can be used for authentication of a communication channel but cannot protect an image posted on a web page.

The solution is to put a robust, secure, invisible watermark on the image and the watermarked image W is distributed. The author keeps the original image I. To prove that an image W' or a portion of it has been pirated, the author shows that W' contains his watermark (to this purpose, he could but does not have to use his original image I). The best a pirate can do is to try to remove the original watermark (which is impossible if the watermark is secure), or he can embed his signature in the image. But this does not help him too much because both his “original” and his watermarked image will contain the author’s watermark (due to robustness property), while the author can present an image without pirate’s watermark. Thus, the ownership of the image can be resolved in the court of law.

Requirements: The watermark must be robust, secure, invisible, and it has to depend in a non-invertible manner on the original image. The watermarking technique can use the original image for watermark detection. This simplifies image registration before watermark detector can be applied. Other requirements: relatively small capacity (1-100 bits).

Fingerprinting (traitor-tracing)

Movies are distributed to different people. One wants to identify those that make illegal copies and sell them. Other scenario includes distributing sensitive information (images, videos) to several deputies and trying to trace down a traitor who leaks information to the enemy. One cannot use visible (audible) marking because such would look suspicious and could be easily removed. The marks must be perceptually invisible and must be present in every frame or image that is being distributed. The marks must be embedded in a robust way so that multiple copying or editing cannot remove them.

Requirements: Robust, secure, invisible watermark that depends in a non-invertible manner on the original is embedded in the document. The watermarking technique should not use the original for watermark detection. Since possibly a large number of documents marked with different marks will be distributed in the public domain, the technique must be resistant with respect to the collusion attack (see section 2.3.2). Other requirements: relatively small capacity.

Adding captions to images, additional information to videos

Movie dubbing in multiple languages, subtitles, tracking the use of the data (history file). For example, one copy of a movie can be distributed with subtitles in several languages. The VCR, DVD player, TV set, or other video device can access and decode the additional text (subtitles) in real time from each frame, and display it on the TV screen. Although this could be arranged by appending information rather than invisibly embedding it, bandwidth requirements and necessary format changes may not allow us to do so.

Requirements: A moderately robust, most importantly with respect to lossy compression, and noise adding, transparent watermark with moderate to large capacity. The original images (frames) are not available for message extraction. Since the hidden information is beneficial for the consumer, there is no need to require security - the consumer is not motivated to remove the hidden information. Since the watermark must be recovered in real time, fast detection is necessary. On the other hand, watermark embedding can be more time consuming.

2.3 Attacks on Watermarks

2.3.1 The IBM attack

Invertible non-oblivious watermarking schemes are vulnerable to this attack. Let an image I_A is formed by a person A by adding a watermark W_A to $I : I_A = I+W_A$. Now, person B generates his watermark W_B using his key and creates a fake original $I' = I_A - W_B$. Since $I_A = I + W_A = I' + W_B$ and since the watermarking method must be robust with respect to small changes, person B can argue that original image I contains his watermark W_B if he uses his forged original I' for the detection. Of course, person A can claim that his watermark W_A is contained in I' if I is used as the original image. This creates a deadlock and one cannot unambiguously decide who owns the image. This attack can be defeated by making the watermark W depend on the original image in a non-invertible manner. In order to forge an original and a watermark, an attacker would have to solve the equation $I_A = I' + W(I')$ for I', which may be computationally very difficult if W for example depends on image hash. In some circumstances, if multiple watermarked copies are available, an attack can still be mounted. Oblivious schemes are also vulnerable to this attack.

2.3.2 Collusion attack

The collusion attack becomes relevant whenever one image is watermarked with different watermarks and those copies are distributed. For example, fingerprinting movies with the intention to identify the customer requires marking the frames with a different watermark that is unique to the customer. If many customers average together the frames from their movies, the watermarks will cancel out and a non-watermarked copy could be obtained. This is called the collusion attack. In some situations, the collusion attack is not relevant simply because only one watermarked copy of a digital image will be ever made and distributed. But for fingerprinting and traitor tracing, the collusion attack needs to be taken into account.

2.3.3 StirMark

This powerful attack has been designed by a research group (R. Andersson[1], F. Petitcolas, and M. Kuhn) at University of Cambridge. The attack simulates image distortions that commonly occur when a picture is printed, photocopied, and rescanned. The image is slightly stretched and compressed by random amounts, a small amount of noise is added to simulate quantization errors of A/D and D/A conversion. The strongest part of this attack is the small geometrical change. They cause loss of synchronization between watermark detector and the image. For low-frequency watermarks, small geometrical deformations can cause large differences in DCT coefficients. StirMark does not pose any threat to non-oblivious watermarking because the original image can be used for registration of the watermarked/attacked image. Oblivious watermarking methods, however, may be seriously disturbed by this attack. Currently, there is no oblivious watermarking scheme that would be able to withstand the StirMark attack.

2.3.4 The mosaic attack

This attack[1] was motivated by an automatic system for copyright piracy detection - a special program (a crawler) that will search through the Internet, download pictures, and look for illegal copies of images watermarked with a certain watermark. The idea behind the mosaic attack is to simply break an image into small portions and correctly assemble them on a web page so that a complete image without spaces is obtained. This is easily done because most browsers can paste images without any spaces in between them. Because images with dimensions smaller than a certain limit cannot be reliably watermarked, the crawler would not detect the watermark in any mosaic piece. Another possibility is to wrap images into Java applets, Active X objects so that they would not be recognized as images to the crawler. The applet can even descramble the image in real time.

This attack can only be overcome by a system that would render the complete web page on a computer screen, detect the images, and search for watermarks in them.

2.3.5 Attack based on partial knowledge of the watermark

It is important that a partial knowledge of the watermark should not enable a pirate to remove the entire watermark or disturb it beyond reliable detection. It might indeed be possible in certain cases to reconstruct the watermark pattern based on the assumption that the watermark becomes partially known. This assumption is not that unreasonable as it may seem at first. For example, one can make a guess that certain portion of the original image had pixels of uniform brightness or of a uniform gradient, or an attacker may be able to foist a piece of his image into a collage created by somebody else. If this is the case, then the knowledge of a portion of the watermark pattern may give us additional constraints to disturb or eliminate the whole watermark. This is especially relevant for watermark patterns spanned by publicly known functions.

2.4 Image Encoding Techniques

Information can be hidden in images in different ways. Straight message insertion can be done, which will simply encode every bit of information in the image. More complex encoding can be done to embed the message only in "noisy" areas of the image, that will attract less attention. The message may also be scattered randomly throughout the cover image.

The most common approaches[2] to information hiding in images are:

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

Each of these can be applied to various images, with varying degrees of success. Each of them suffers to varying degrees from operations performed on images, such as cropping, or resolution decrementing, or decreases in the colour depth.

2.4.1 Least Significant bit insertion

The least significant bit insertion method[3] is a common, simple approach to embedding information in a graphical image file. Unfortunately, it is extremely vulnerable to attacks, such as image manipulation. A simple conversion from a GIF or BMP format to a lossy compression format such as JPEG can destroy the hidden information in the image.

When applying LSB techniques to each byte of a 24-bit image, three bits can be encoded into each pixel (as each pixel is represented by three bytes). Any changes in the pixel bits will be indiscernible to the human eye. When using LSB techniques on 8-bit images, more care needs to be taken. A change of even one bit could mean the difference between a shade of red and a shade of blue. So grey-scale palettes are recommended for data hiding. This increases the image's ability to hide information.

2.4.2 Masking and filtering

Masking and filtering techniques hide information by marking an image in a manner similar to paper watermarks. Because watermarking techniques are more integrated into the image, they may be applied without fear of image destruction from lossy compression. By covering, or masking a faint but perceptible signal with another to make the first non-perceptible, we exploit the fact that the human visual system cannot detect slight changes in certain temporal domains of the image.

Technically, watermarking is not a steganographic form. Strictly, steganography conceals data in the image; watermarking extends the image information and becomes an attribute of the cover image, providing license, ownership or copyright details.

Masking techniques are more suitable for use in lossy JPEG images than LSB insertion because of their relative immunity to image operations such as compression and cropping.

2.4.3 Algorithms and transformations

Transformations: JPEG images use the discrete cosine transform (DCT) to achieve compression. DCT is a lossy compression transform, because the cosine values cannot be calculated precisely, and rounding errors may be introduced. Variances between the original data and the recovered data depends on the values and methods used the calculate the DCT[4].

Images can also be processed using fast Fourier transformation and wavelet transformation. Other properties such as luminance can also be utilised. The HVS has a very low sensitivity to small changes in luminance, being able to notice changes of not less than one part in thirty for random patterns and one part in 240 for uniform regions of an image.

Spread-spectrum techniques: Modern steganographic systems use spread-spectrum communications[5] to transmit a narrowband signal over a much larger bandwidth so that the spectral density of the signal in the channel looks like noise.

The two different spread-spectrum techniques these tools employ are called direct-sequence and frequency hopping. The former hides information by phase-modulating the data signal (carrier) with a pseudorandom number sequence that both the sender and the receiver know. The latter divides the available bandwidth into multiple channels and hops between these channels (also triggered by a pseudorandom number sequence).

Patchwork method: Bender, Gruhl, and Morimoto introduce a technique called patchwork [6]. Pairs of pixels A and B with gray levels a and b are randomly chosen in the image. The expected value of the difference $A - B$ is zero. Repeating this procedure n times, we can form the quantity S

$$S = \sum_{i=1}^{n} (a_i - b_i) \tag{2.1}$$

The expected value of this sum is zero with variance $\sigma^2 = 10922.5 \times n$ (for 256 gray levels). Using statistics, it is possible to estimate the probability that the value of S will exceed certain threshold. The watermarking algorithm starts with a secret key used to generate random numbers. The sequence of pseudo-random numbers is used to randomly select n pixel pairs. For each pair, the gray level of one of the first pixel is increased by one, while the value of the second pixel is decreased by one. To prove that an image is watermarked with a specific secret key, one generates the sequence of pseudo-random numbers and evaluates the sum S. Again, hypotheses testing can be used to confirm the presence of a watermark on a certain confidence level. The method can be improved by adjusting patches of pixels rather than single pixels. This will have the effect of shifting the energy of the watermark towards low frequencies thus making it more robust to JPEG compression and low-pass filtering. As with most spread spectrum methods, the watermark is very robust to nonlinear transformations of the gray scale.

Chapter 3

Fractals and Image Compression

3.1 What are Fractals ?

Fractals were first described in the 1970s by IBM mathematician Benoit Mandelbrot. He found traditional geometry to be incomplete. It could not describe the enormous and irregular shape of a mountain. It had no formal representation of the appearance of a cloud. The new geometry that he developed could do all this. It was a description of the beautiful yet irregular and fragmented patterns around us.

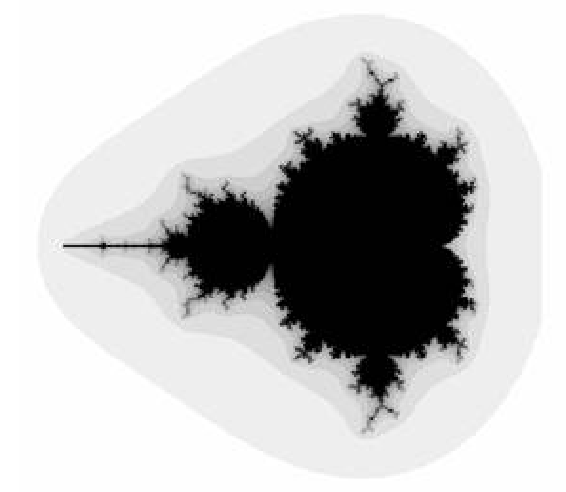


Figure 3.1: Fractal generated with the Mandelbrot set. The boundary of the black interior exhibits infinite detail and self-similarity.

The term 'fractal' was coined by Mandelbrot from the Latin fractus, an adjective for the irregular and the fragmented. Essentially, they replicate themselves by fragmentation. Mandelbrot set is shown in figure 3.1. If we look at a fractal pattern and we see a form; then we look closely at a particular region of the pattern, and we will see the same form all over again, only much smaller this time. And so on it goes, from the largest scales to the smallest. So the fractals are the repetition of the same structural form.

3.2 Iterated Function Systems

Imagine a special type of photocopying machine that reduces the image to be copied by a half and reproduces it three times on the copy. Figure 3.2 shows this. The output of this machine is fed back as input. Figure 3.3 shows several iterations of this process on several input images. All the copies seem to be converging to the same final image. This image is called the attractor for this copying machine. Because the copying machine reduces the input image, any initial image will be reduced to a point as we repeatedly run the machine. Thus, the initial image placed on the copying machine doesn't effect the final attractor. In fact, it is only the position and the orientation of the copies that determines what the final image will look like.

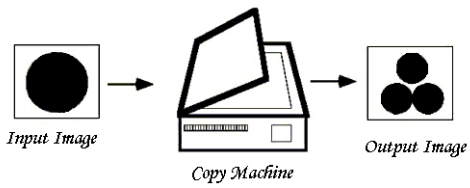


Figure 3.2: A copy machine that makes three reduced copies of the input image.

Different transformations lead to different attractors, with the technical limitation that the transformations must be contractive - that is, a given transformation applied to any two points in the input image must bring them closer together in the copy. This technical condition is very natural, since if points in the copy were spread out the attractor would have to be of infinite size. Except for this condition, the transformations can have any form. In practice, choosing transformations of the form

$$w_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}$$

is sufficient to yield a rich set of attractors. Such transformations are called affine transformations of the plane, and each can skew, stretch, rotate, scale and translate an input image; in particular, affine transformations always map squares to parallelograms. A collection w_1, w_2, ..., w_n of contractive maps is called *iterated function system*(IFS).

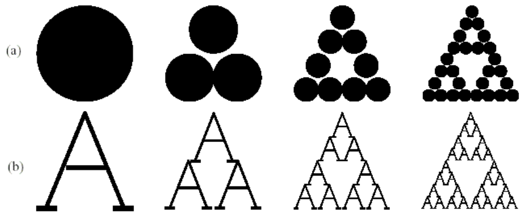


Figure 3.3: The first three copies generated on copying machine of previous figure.

Figure 3.4 shows some affine transformations, the resulting attractors, and a zoom on a region of the attractor. The transformations are displayed by showing an initial square marked with an "⌞" and its image by the transformations. The "⌞" helps show when a transformation flips or rotates a square. The first example(Sierpinski triangle) shows the transformations used in the copy machine of figure 3.4. These transformations reduce the square to half its size and copy it at three different locations in the same orientation. The second example is very similar to the first, but in it, one transformation flips the square resulting in a different attractor. The last example is the Barnsley fern. It consists of four transformations, one of which is squished flat to yield the stem of the fern.

A common feature of these and all attractors formed this way is that in the position of each of the images of the original square on the left there is a transformed copy of the whole image. Thus, each image is formed from transformed (and reduced) copies of itself, and hence it must have detail at every scale. That is, the images are fractals.

3.3 Partition Iterated Function System

Most fractal coding schemes are based on representation by Partition Iterated Function System(PIFS), a solution to the inverse problem which was first published by Jacquin[7]. A PIFS differs from an IFS in the individual mappings operate on a subset of image, rather than the entire image.

3.3.1 Self-similarity in Natural Images

An image of a face in figure 3.5(a) does not contain self-similarity that can be found in the fractals in figure 3.3. The image does not appear to contain affine

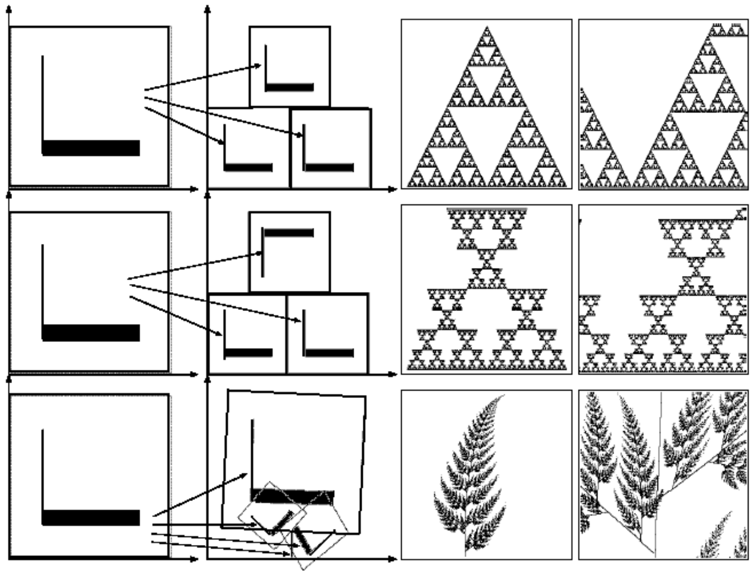


Figure 3.4: Transformations, their attractor, and a zoom on the attractors.

transformations of itself. But, in fact, this image does contain a different sort of self-similarity. Figure 3.5(b) shows sample regions which are similar at different scales: a portion of shoulder overlaps a region that is almost identical, and a portion of the reflection of the hat in the mirror is similar (after transformation) to a part of hat. The distinction from the kind of self-similarity in figure 3.3 is that rather than having the image be formed of copies of its whole self (under

Figure 3.5: (a) Original 256×256 pixel Lenna image. (b) Self similar portions of image.

appropriate affine transformation), here the image will be formed of copies of properly transformed parts of itself. These transformed parts do not fit together, in general, to form an exact copy of the original image, and so we must allow some error in our representation of an image as a set of transformations. This means that the image we encode as a set of transformations will not be an identical copy of the original image but rather an approximation of it.

Each transform w_i that operates only on a sub-region of the image is referred to as domain blocks D_i. The image sub-regions to which the domain blocks are mapped are called range blocks R_i. In coding of greyscale images, the image is represented as function the Euclidean plane, where the height of the surface at each point represents the local pixel intensity.

$$w_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}$$

where (x,y) denotes the coordinates of a point in the image and z= f(x,y) denotes the intensity or grey level at (x,y). Each transformation w_i can be split into two transformations L_i and T_i, defined as

$$L_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}$$

and

$$T_i(z) \quad = \quad s_i z + o_i.$$

L_i determines how the partitioned domain is mapped to the range, while s_i and o_i determine the contrast and brightness of the transformation. The transformation w_i and domain blocks D_i are chosen such that the error, given by

$$\varepsilon_i \quad = \quad \int\int_{R_i} [f(x,y) - T_i(f(L_i^{-1}(x,y)))]^2 dxdy$$

is minimum.

3.3.2 Ways to Partition Images

The first decision to be made when designing a fractal coding scheme is the choice of the type of image partition used for the range blocks. Partition the image by some collection of ranges R_i. Then for each R_i seek from some collection of image pieces a D_i which has a low rms error. The sets R_i and D_i, determine s_i and o_i as well as a_i, b_i, c_i, d_i, e_i and f_i. We then get a transformation $\mathrm{W} = \bigcup w_i$ which encodes an approximation of the original image.

Fixed square blocks

The simplest range partition consists of the fixed square blocks. Fixed block partitions, however, strongly neglect the image content. As a consequence, it leads to severe blocking artifacts and a rapidly deteriorating image quality. Novel coding concepts thus try to abandon fixed block partitions in favor of a content related picture representation.

The usual solution is to introduce an adaptive partition with large blocks in low detail regions and small blocks where there is significant detail.

Quadtree Partitioning

The quadtree partition[8] employs the well-known image processing technique based on a recursive splitting of selected image quadrants, enabling the resulting partition to be represented by a tree structure in which each non-terminal node has four descendants. The partition is constructed by selecting an initial level in the tree (corresponding to some maximum range block size) and recursively partitioning any block for which a match better than some preselected threshold is not found. Compact coding of partition details is possible by taking advantage of the tree structure of the partition.

HV-Partitioning

A weakness of the quadtree based partitioning is that it makes no attempt to select the block in a content dependent way. The collection must be chosen to be very large so that a good fit to a given range can be found. A way to remedy this, while increasing the flexibility of the range partition, is to use an HV-partition. In an HV-partition, a rectangular image is recursively partitioned either horizontally or vertically to form two new rectangles. The partitioning repeats recursively until a covering tolerance is satisfied, as in the quadtree scheme.

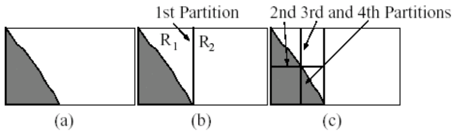


Figure 3.6: The HV scheme attempts to create self similar rectangles at different scales.

This scheme is more flexible, since the position of the partition is variable. The partitions can be made in such a way that they share some self similar structure. For example, we can try to arrange the partitions so that edges in the image will tend to run diagonally through them. Then, it is possible to use the larger partitions to cover the smaller partitions with a reasonable expectation of a good cover. Figure 3.6 demonstrates this idea. The figure shows a part of an image (a); in (b) the first partition generates two rectangles, R1 with the edge running diagonally through it, and R_2 with no edge; and in (c) the next three partitions of R_1 partition it into 4 rectangles, two rectangles which can be well covered by R_1 (since they have an edge running diagonally) and two which can be covered by R_2 (since they contain no edge).

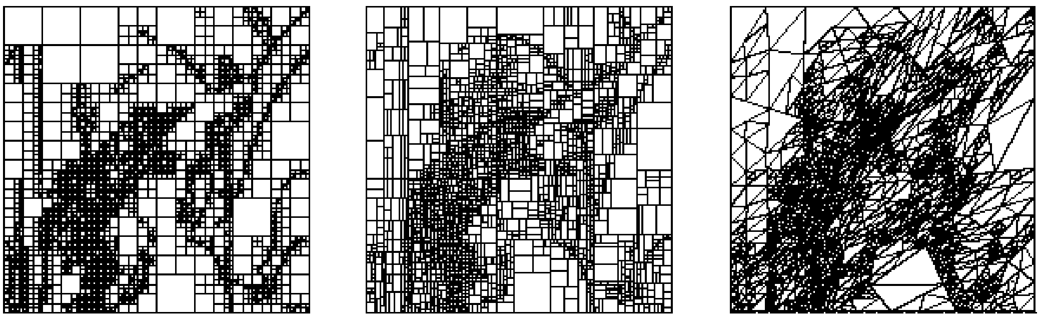


Figure 3.7: A quadtree partition (5008 squares), an HV partition (2910 rectangles), and a triangular partition (2954 triangles).

Triangular Partitioning

Another way to partition an image is based on triangles. In the triangular partitioning scheme, a rectangular image is divided diagonally into two triangles. Each of these is recursively subdivided into 4 triangles by segmenting the triangle along lines that join three partitioning points along the three sides of the triangle. This scheme has several potential advantages over the HV-partitioning scheme. It is flexible, so that triangles in the scheme can be chosen to share self-similar properties, as before.

Figure 3.7 shows sample partitions arising from the three partitioning schemes applied to the Lenna image.

Polygonal

Polygonal partition is constructed by recursively subdividing an initial coarse grid. Each polygon is allowed to be subdivided at an arbitrary position in the polygon by the insertion of a line segment at one of a restricted set of angles, thus reducing the information required in coding the partition details.

3.3.3 Search Strategies

The significant computational requirements of the domain search resulted in lengthy coding times for early fractal compression algorithms. The design of efficient domain search techniques has consequently been one the most active areas of research in fractal coding, resulting in a wide variety of solutions.

Domain Pool Restriction

The first step in the reduction of computational complexity of the encoding process is to restrict the domain pool. Not all possible domain blocks are searched. One restriction on domain pool could be that the block size is twice the size of the range blocks, with adjacent domain blocks overlapping one another by the size of the range blocks. Search for the appropriate domain block is carried out in a spiral path around the range block.

Block Classification

Jacquin used a classification scheme in which domain and range blocks are classified as shade, edge and mixed blocks. Edge blocks are further classified as simple and mixed edge blocks. For a range block from any one of the above categories, only domain blocks from the same category are searched.

A more elaborate classification scheme was proposed by Fisher et al[9]. If A_i, $i = 1...4$ are the mean pixel intensities of the four quadrants of a range or domain block, then we can always orient the block (by flipping or rotation) so that the block falls into one of the following three major classes:

- Major Class 1: $A_1 \geq A_2 \geq A_3 \geq A_4$
- Major Class 2: $A_1 \geq A_2 \geq A_4 \geq A_3$

- Major Class 3: $A_1 \geq A_4 \geq A_2 \geq A_3$

Further a block belonging to one particular major class can be classified into $4! = 24$ ways depending on the ordering of the variances of each quadrant. Thus there are effectively 72 classes of blocks. For a range block belonging to one particular class, only domain block of the same class is searched.

Adaptive Clustering

In the methods described earlier, the classification scheme is decided upon before it is applied to any image. On the other hand, in the adaptive clustering scheme, the classification is image dependent. In this method, the range and code book blocks are divided into disjoint sets around 'cluster centers'.

Feature Vectors

In this method a small set of d real-valued keys are assigned to each range and domain block, which make up the d-dimensional *feature vector*. These keys are constructed such that searching in the domain pool is restricted to a small neighborhood around the d-dimensional key corresponding to a particular range block. Thus the sequential search of the domain block is substituted by a nearest neighbor search.

3.4 Compression

To encode the image, it is partitioned into blocks. We need to select an image-partitioning scheme to generate range blocks R_i. It is convenient to select domains with twice the range size and to subsample or average groups of 2×2 pixels to get a reduced domain with the same number of pixels as the range. For each range block, a best matching domain block is searched. The pixels in the domain are averaged in the groups of four so that the domain is reduced to size of the range, and the affine transformation of the pixel values (that is, a scaling and offset) is found that minimizes the rms difference between the transformed domain pixel values and the range pixel values.

For a fixed image, more transformations lead to better fidelity but worse compression. This trade-off between compression and fidelity leads to two different approaches to encoding an image - one targeting fidelity and one targeting compression. These approaches are outlined in the pseudo-code in Tables 3.1 and 3.2

```
• Choose a tolerance level e_c.
• Mark all ranges R_i uncovered.
• While there are uncovered ranges R_i do {
      • Out of possible domains D, find the domain D_i and the
      corresponding w_i that best cover R_i (i.e.  that minimize RMS).
      • If RMS < e_c or size(R_i) ≤ r_min then
            • Mark R_i as covered, and write out the transformation w_i.
      • else
            • Partition R_i into smaller ranges that are marked as
            uncovered, and remove R_i from the list of uncovered ranges.
}
```

Table 3.1: Pseudo-code targeting a fidelity e_c.

```
• Choose a target number of ranges N_r.
• Set a list to contain all ranges R_i, and mark it as uncovered.
• While there are uncovered ranges in the list do {
      • For each uncovered range in the list, find and store the
      domain D_i ∈ D
      • Out of the list of ranges, find the range R_j with size(R_j)
      > r_min with the largest RMS (i.e.  which is covered worst).
      • If the number of ranges in the list is less than N_r then {
            • Partition R_j into smaller ranges which are added to the
            list and marked as uncovered.
            • Remove R_j, w_j and D_j from the list.
      }
}
• Write out all the w_i in the list.
```

Table 3.2: Pseudo-code targeting an encoding with N_r transformations.

3.5 Decompression

Decompression of an image is simple. Starting from any initial image, we repeatedly apply the w_i until we approximate the fixed point. This means that for each w_i, we find the domain D_i, shrink its size to the range R_i, multiply the pixel values by s_i and add o_i and put the resulting pixel values in the position of R_i. This constitutes one decoding iteration. The decoding step is iterated until the fixed point is approximated, that is, until further iteration doesn't change the image or until the change is below some small threshold value. Typically, 10 iterations are sufficient.

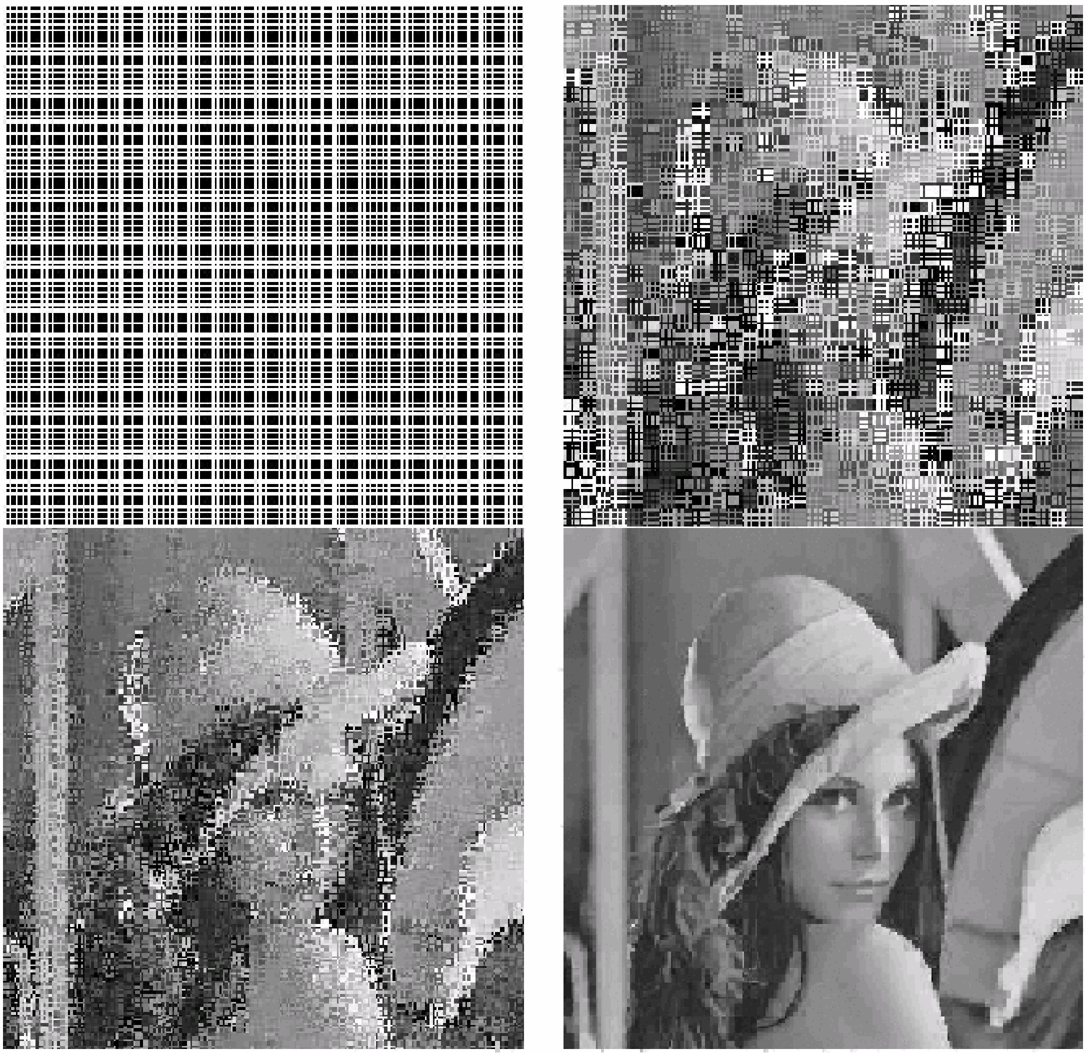


Figure 3.8: The initial image, and the first, second, and tenth iterates of the encoding transformations.

Chapter 4

Fractal Based Information Hiding

4.1 Basic Idea

Our scheme hides the data in spatial domain. This method takes an image file and steganographic data and produces a new image file that contains the steganographic data. The output image is called steganographic image file and it is similar to the input image file. Fractal encoding technique identifies parts of the image that are most suited for data hiding.

4.1.1 Hiding Steganographic Data

Image is partitioned in to regions: domain region D and range region R. A key is given by the user. Data which is to be hidden is XOR with key before hiding it. The sub-images within the range region and domain region are called range blocks and domain blocks, respectively.

A domain library is build from a set of blocks in the domain region. It is split into two halves D^0 and D^1. Fractal image compression theory is used to identify a series of range blocks and a corresponding series of matching domain blocks. The matching process produces for each range block, the best matching domain block and a corresponding fractal transform.

The fractal transform basically involves multiplying a pixel in the selected domain block by an average scaling factor and adding an offset factor to the result. Thus the fractal transformation is applied to the domain block to produce a new range block that is visually similar to the original range block. This new block is written to the steganographic image in place of the old range block.

If the current bit of steganographic data is 0, we search for a match in the first half of the domain library otherwise we search the second half. For each bit of steganographic data a new range block is produced.

4.1.2 Retrieving Steganographic Data

Retrieving steganographic method is reverse of hiding. Range and domain regions remain same as in hiding process. To find a bit of steganographic data in the image file, we select a range block from the range region in the same order as in the hiding process and find corresponding domain block that matches it by a linear relationship. The linear relationship is that each pixel in range block is formed by multiplying corresponding pixel in the domain block by a scale factor and adding an offset factor. We can not use fractal image compression techniques for the match as relationship between the two blocks is linear. If we use fractal image compression techniques, they would also yield some blocks that are similar but have not been used in data hiding. So, for matching we compute scaling and offset factors from any two pixels in the range and domain blocks, then they are applied on rest of the pixels in the domain block and determined if the corresponding pixel exists in the range block.

The match is successful if we can find this linear relationship between the two blocks. If match occurs in D^0, the first half of domain region, we set current bit of steganographic data to 0, otherwise if match occurs in D^1, we set the bit to 1. Finally, this sequence of bits is XOR with key to get the original data back.

4.2 Related Work

Fractal encoding can be done in spatial domain[10][11] and frequency domain[12]. Paul Darvin[13] suggested a fractal method that uses the information of range and domain region as key. Joan Paute and Fred Jordan[14] used key to generate coordinates of range blocks. Patrick Bas and Jean-Marc Chassery[15][16] used fractal scheme for watermarking. They find the transformation such that for each block D and R, a new range block $\hat{R}$ is calculated as

$$\hat{R} = \delta * S * \frac{D}{max(D)} + \bar{R}$$

where S determines the magnitude of the watermark, $\bar{R}$ is the mean of R and

$$\delta = \begin{cases} +1 \text{ if the embedded bit=1} \\ -1 \text{ if the embedded bit=0} \end{cases}$$

4.3 Implementation

Implementation of the algorithm is based on the following assumptions:

- Append data with the label `END_OF_DATA`.
- XOR data with key.
- Subsample domain blocks D so that they have same number of pixels as range blocks.
- Classify all domain blocks.
- While there are bits to be stored {
    - Take a range block R_i from R.
    - Classify R_i.
    - If current bit b_i is 0
        - Consider domain blocks from D^0 that are of same class as of R_i.
    - else
        - Consider domain blocks from D^1 that are of same class as of R_i.
    - Calculate scale s_i, offset o_i and rms distance d_{rms} between R_i and all D_js in the selected quadrant.
    - Select domain block D_{min} with least d_{rms} and corresponding s_i and o_i.
    - Multiply each pixel of D_{min} by s_i and add o_i and overwrite the corresponding range pixel by the result.

}

Table 4.1: Pseudo-code for hiding steganographic data.

- While not `END_OF_DATA` {
    - Take a range block R_i from R.
    - Calculate scale s_i, offset o_i and rms distance d_{rms} between R_i and all D_js in D.
    - Select domain block D_{min} with least d_{rms} and corresponding s_i and o_i.
    - If D_j is in D^0
        - Data bit b_i is 0.
    - else
        - Data bit b_i is 1.

}
- XOR this bit stream with key to get original data.

Table 4.2: Pseudo-code for retrieving steganographic data.

- It is assumed that the input image is of equal length and breadth.
- Image is grey scale and in the sun-raster format.
- Size of image header is 800 bytes.
- Each byte in the image data represents a pixel whose level of grey is from 0 to 255.

The hiding algorithm takes image I, steganographic data and a key as input. It outputs a visually identical image I_{new} that has steganographic data hidden in it. The retrieving algorithm takes image I and key as input and outputs steganographic data.

Image I is partitioned into four quadrants. First and third quadrants constitute the range region R and second and fourth quadrants domain region D. Range region is divided into square blocks $\{r_0, r_1, \ldots, r_l\}$ of equal size (4×4 or 8×8). These blocks are non-overlapping.

A domain library is built from a set of blocks in the domain region. The length of both sides of domain block is twice that of range blocks therefore there are 4 times as many pixels in a domain block than there are in a range block. Domain blocks are overlapping. Domain library D is represented by the blocks $\{D_0, D_1, \ldots, D_m\}$, so m+1 is the number of domain blocks in D. Domain library is split into two halves $\{D_0, D_1, \ldots, D_{m/2}\}$ which are from first quadrant of image and $\{D_{(m/2)+1}, D_{(m/2)+2}, \ldots, D_m\}$ from third quadrant of image. First half of D is D^0 and other is D^1.

We used fixed block partition. Size of all range blocks and domain blocks is fixed. The search strategy we used is block classification as described in section 3.3.3.

Domain and range blocks are compared using rms metric. Given two squares containing n pixel intensities, $a_1, \ldots, a_n$ (from D_i) and $b_1, \ldots, b_n$ (from R_i), we can seek s and o to minimize the quantity

$$R' = \sum_{i=1}^{n} (s \cdot a_i + o - b_i)^2. \tag{4.1}$$

This will give us contrast and brightness settings that make the affinely transformed a_i values have the least squared distance from the b_i values. The minimum of R' occurs when the partial derivatives with respect to s and o are zero, which occurs when

$$s = \frac{\left[n \sum_{i=1}^{n} a_i b_i - \sum_{i=1}^{n} a_i \sum_{i=1}^{n} b_i \right]}{\left[n \sum_{i=1}^{n} a_i^2 - \left(\sum_{i=1}^{n} a_i \right)^2 \right]} \tag{4.2}$$

and

$$o = \frac{1}{n} \left[\sum_{i=1}^{n} b_i - s \sum_{i=1}^{n} a_i \right] \tag{4.3}$$

In that case,

$$R' = \frac{1}{n} \left[\sum_{i=1}^{n} b_i^2 + s \left(s \sum_{i=1}^{n} a_i^2 - 2 \sum_{i=1}^{n} a_i b_i + 2o \sum_{i=1}^{n} a_i \right) + o \left(no - 2 \sum_{i=1}^{n} b_i \right) \right] \tag{4.4}$$

The rms error is equal to $\sqrt{R'}$.

4.3.1 Method 1

Input image is partitioned into range and domain regions and sub-blocks as described earlier. Range blocks are selected in a pre-defined order. Then the matching domain block is searched in the I or III quadrant of the image depending upon the bit value which to be stored. Before comparing a domain block D_j with the range block R_i, D_j is contracted by a factor of two on each side by averaging neighboring pixels (see figure 4.1), followed by the application of one of the eight rotations and reflections (see figure 4.2) making up the isomerties of a square (see figure 4.3). This increases the size of domain library by eight times, so there are greater chances of getting a good match. For each R_i, we calculate s_i, o_i and

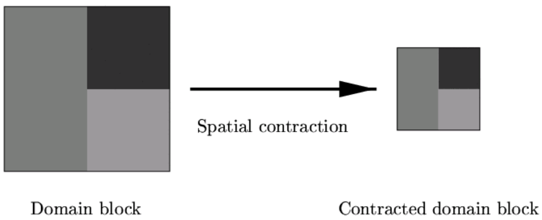


Figure 4.1: Spatial contraction of a domain block.

rms value corresponding to all D_js in the selected quadrant. These parameters are calculated as given in equations 4.2, 4.3 and 4.4 respectively. Domain block corresponding to which rms value is minimum, is chosen alongwith s_i and o_i. Each

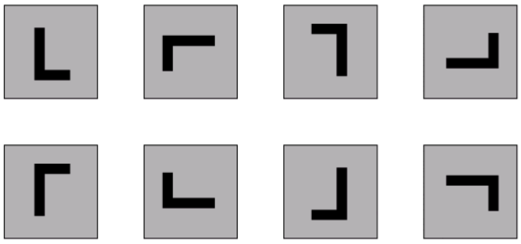


Figure 4.2: The square isometries.

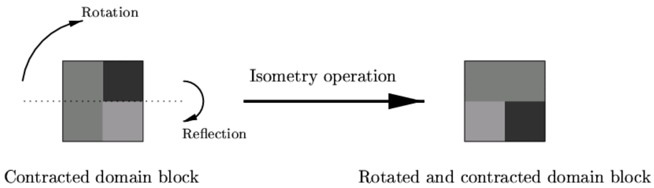


Figure 4.3: An isometry applied to a domain block.

pixel of the subsampled domain block is multiplied by s_i and added to o_i to generate a new range block $R_{i_{new}}$ (see figure 4.4) which is written over R_i. So, the bits are stored in form of mappings from domain region to range region. The number of such mappings is equal to the number of bits stored and hence, equal to the number of range blocks modified.

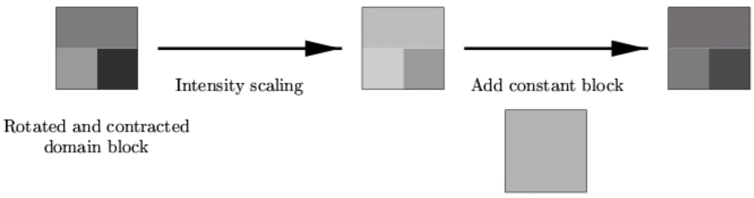


Figure 4.4: An affine transform applied to a domain block.

While extracting the message, range blocks are selected in the same order as in hiding process. Here, we consider all domain blocks for each R_i. We calculate s_i and o_i (from equations 4.2 and 4.3) which are applied on D_j and rms value is calculated from equation 4.4 for R_i. In this way, rms value is calculated between R_i and all D_js in I and III quadrants. If the domain block D_j with minimum rms value is in I quadrant, bit value b_i is 0 else if D_j is in III quadrant bit value b_i is 1. Figure 4.5 shows mappings from domain region to range region which store the bit string 100.

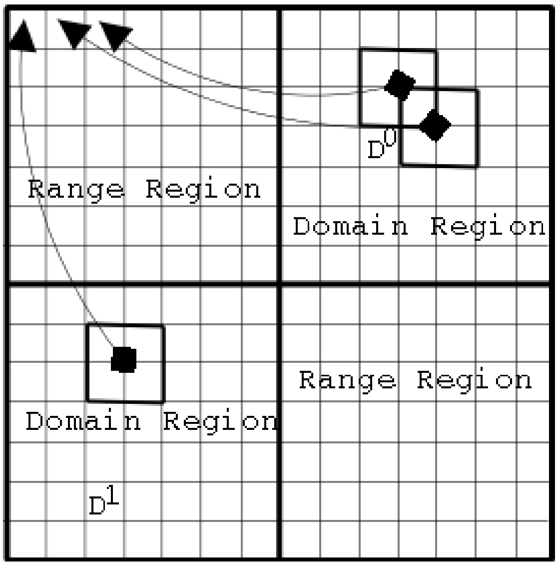


Figure 4.5: Mappings from domain region to range region.

4.3.2 Method 2

For a range block R_i, we calculate s_i, o_i using the equations 4.2 and 4.3 and find the best matching domain block (for which the rms is minimum) in the same way as in previous method. Apply s_i and o_i on the first pixel a_1 of domain block to produce new range pixel $b_{1_{new}}$. Now, apply s_i and o_i to the next pixel $a_k \in \{a_2, a_3, \ldots, a_n\}$ producing a value $b_{k_{new}}$. This new value $b_{k_{new}}$ should not be equal to $b_{1_{new}}$. If they are equal we take the next pixel of domain block and produce new $b_{k_{new}}$ until that is not equal to $b_{1_{new}}$. If $b_{k_{new}} \notin \{0,1,\ldots,255\}$ we skip this pixel. We now calculate $s_{i_{new}}$ and $o_{i_{new}}$ as follows:

$$s_{i_{new}} = \frac{a_k - a_1}{b_{k_{new}} - b_{1_{new}}} \tag{4.5}$$

$$o_{i_{new}} = a_k - s_{i_{new}} \cdot b_{k_{new}} \tag{4.6}$$

$b_{1_{new}}$ and $b_{k_{new}}$ in the equations 4.5 and 4.6 are integers. Their fractional part is ignored. Apply $s_{i_{new}}$ and $o_{i_{new}}$ to $\{a_{k+1}, a_{k+2}, a_{k+3}, \ldots, a_n\}$ producing $\{b_{(k+1)_{new}}, b_{(k+2)_{new}}, b_{(k+3)_{new}}, \ldots, b_{n_{new}}\}$. Now, we write only $b_{1_{new}}, b_{(k+1)_{new}}, b_{(k+2)_{new}}, \ldots, b_{n_{new}}$ over the corresponding pixels of R_i.

In the retrieving process, for each selected range block R_i we examine all domain blocks. We find b_1, b_k, a_1 and a_k such that $b_1 \neq b_k$. s_i and o_i are calculated using the equations 4.5 and 4.6. These parameters are applied on the pixels $\{a_{k+1}, a_{k+2}, \ldots, a_n\}$ of D_j and compared with the pixels $\{b_{k+1}, b_{k+2}, \ldots, b_n\}$ of R_i. Domain block is selected with the minimum difference from R_i and according to the location of domain block bit is decoded.

While hiding, if the fractional part of $b_{1_{new}}$ and $b_{k_{new}}$ is not ignored then the value of s_i and o_i will be calculated using their fractional part also. But the value of a pixel can not have fractional part, so that will be ignored (or rounded-off) while writing over the range block. So, in retrieving process, s_i and o_i are calculated using only the integer values. Therefore, the value of s_i and o_i calculated while hiding will be different from calculated while retrieving. Due to the different value of s_i and o_i, rms value between the range and domain blocks will change and hence the mapping will change. Change in mapping may lead to the retrieval of incorrect information.

4.3.3 Method 3

Here we consider only the least significant bit (LSB) of the pixels. LSB of the pixels of range block R_i are compared with corresponding LSB of the pixels of all domain blocks in the selected quadrant. Domain blocks D_j with the minimum difference is chosen and its least significant bit plane is copied over least significant bit plane of range block R_i. LSB of the pixels of the new range block is same as the LSB of the pixels of the corresponding domain block. So, the mapping is a identity function which is applied on LSB plane of blocks. In this way, a bit is stored.

To get the message back, range blocks are selected in the same order as in hiding process. For a range blocks R_i, LSB of its pixels are compared with the LSB of the pixels of all domain blocks. Domain block is selected with the minimum difference from R_i and according to the location of domain block bit is decoded.

Figure 4.6: (a) Original Lenna 256×256 image. (b) Lenna image containing data using method 1. (c) Lenna image containing data using method 2. (d) Lenna image containing data using method 3. These results are for 256×256 Lenna image with data length 49 bits and range block size 4×4.

Chapter 5

Results and Analysis

5.1 Gathering Results

The testing of each method was performed on a system with Pentium III 700MHz processor and Linux operating system. All the images were 256×256, 8-bit greyscale, of Lenna. The signal to noise ratio, equation 5.1, is a commonly used pixel-based visual distortion metric and this was used to measure the distortion between the original image and the image containing data. A low SNR means that the image has been greatly distorted.

$$SNR = \frac{\sum_{x,y} P_{x,y}^2}{\sum_{x,y} (P_{x,y} - \tilde{P}_{x,y})^2} \tag{5.1}$$

Where, SNR is the signal to noise ratio,
$P_{x,y}$ is a pixel in the original image with coordinates (x,y), and
$\tilde{P}_{x,y}$ is a pixel in the image containing data with coordinates (x,y).
The signal to noise ratio is usually measured in decibels and converted using equation 5.2.

$$SNR(dB) = 10log_{10}(SNR) \tag{5.2}$$

The percentage accuracy of the methods was calculated by finding the number of correct bits recovered (D_τ) and dividing it by the total number of bits in the actual data (D_A).

$$Accuracy = \frac{D_\tau}{D_A} \times 100 \tag{5.3}$$

5.2 Results of Method 1

In the first experiment a data of 49 bits was hidden into a 256×256 Lenna image and results were taken using different range block size. It was observed that if the size if the range and domain block is increased, signal to noise ratio (SNR) decreases. But for the range block size 2×2, SNR is very low. This is due to the inability to find proper s and o as the range block size is too small. For 4×4 and 8×8 range blocks SNR is good enough. Again for 16×16 range block, SNR is reduced to very low value (see figure 5.2(a)). It decreases as the data length is increased (see figure 5.3(a) and 5.3(b)).

Accuracy of getting data back is very less. It is between 40 to 60 percent (see figure 5.2(b)). This is because while hiding the data, only integer part of the new range block pixel is written over the image. Fractional part is ignored. So, in the retrieving process we don't get back the same value of s and o for a range and domain block pair as calculated while hiding.

Hiding time and retrieving time dropped on increasing the block size (see figure 5.2(c) and 5.2(d)). Hiding time increased sharply with the increase in data length (see figure 5.5), but retrieving time increased slowly (see figure 5.6).

5.3 Results of Method 2

The value of SNR decreased with the increase in the block size. It went down sharply in the beginning, but after range block size 8×8 it decreased slowly (see figure 5.2(a)). Sudden falls in SNR were observed when the data length was increased with range block size 4×4 (see figure 5.3(a)). But with 8×8 it was nearly constant when data length was more than 28 bits (see figure 5.3(b)). Some patch were observed on the image after hiding the data (see figure 5.1). This is because while hiding the data, recalculated parameters s and o are not equivalent to the actual s and o. So, the newly generated range block is not similar to the original range block.

Figure 5.1: Patches on the Lenna image after hiding a data of 49 bits with range block size 8×8.

For the range block size 2×2, data is not correctly retrieved. But for the larger blocks, it works well (see figure 5.2(b)). Hiding time and retrieving time decreased gradually with the increase in block size (see figure 5.2(c) and 5.2(d)). But they increased sharply as the data length is increased (see figure 5.5 and 5.6).

5.4 Results of Method 3

SNR decreased with the increase in block size. As the block size increases, SNR converges to some value for a fixed data (see figure 5.2(a)). It also decreased, as the length of data was increased (see figure 5.3).

This method did not work for block size 2×2, because there were more than one domain block matching with a range block. So, it was likely to have incorrect matching domain block and so incorrect bit value. But for larger block size, accuracy was good (see figure 5.2(b)).

Hiding time and retrieving time decreased with the increase in block size. However, retrieving time was nearly same for range block sizes 8×8 and 16×16 (see figure 5.2(c) and 5.2(d)). As the data length was increased, hiding time and retrieving time were increased very slowly (see figure 5.5 and 5.6). So, for this method, length of data does not effect the time taken for the execution of both hiding and retrieving algorithm.

5.5 Attacks on the Watermarked Image

Several attacks were done on the watermark put by the methods 2 and 3 with range blocks 4×4, 8×8 and 16×16. A 256×256 greyscale Lenna image was watermarked. Length of the watermark was 49 bits. Attacks performed are change of brightness and contrast, addition of noise, JPEG compression and scaling of watermarked image. All these were performed using the "gimp" software.

5.5.1 Changing Brightness and Contrast

We changed the brightness and contrast of the watermarked image by different amount. It was observed that method 2 is robust against this attack. Watermark was correctly extracted when we used method 2 with range block size 16×16. For range blocks 4×4 and 8×8, watermark was retrieved with the accuracy of more than 90% (see table 5.1).

Method 3 is not robust to this attack. However, it has shown slightly better results when brightness and contrast was set to 10 and range block size was 8×8 and 16×16 (see table 5.2).

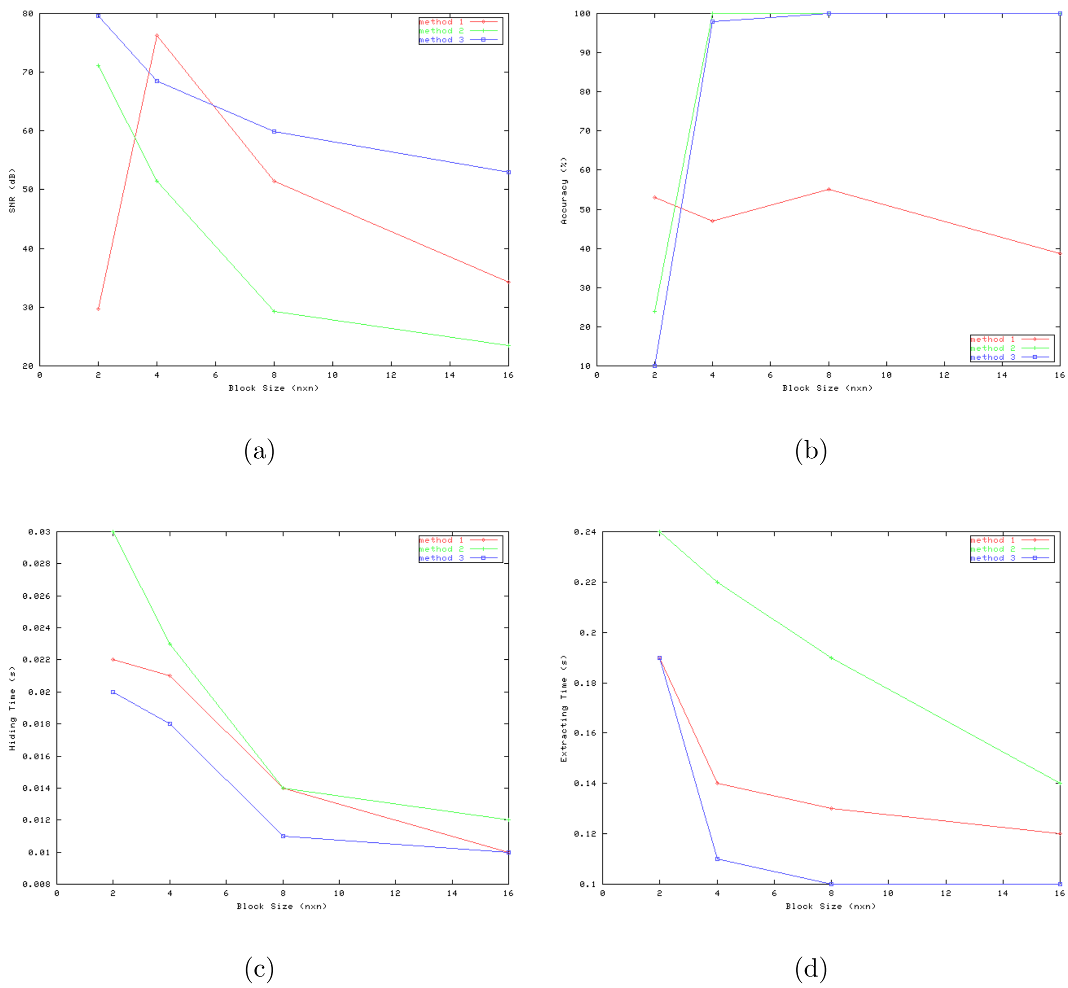
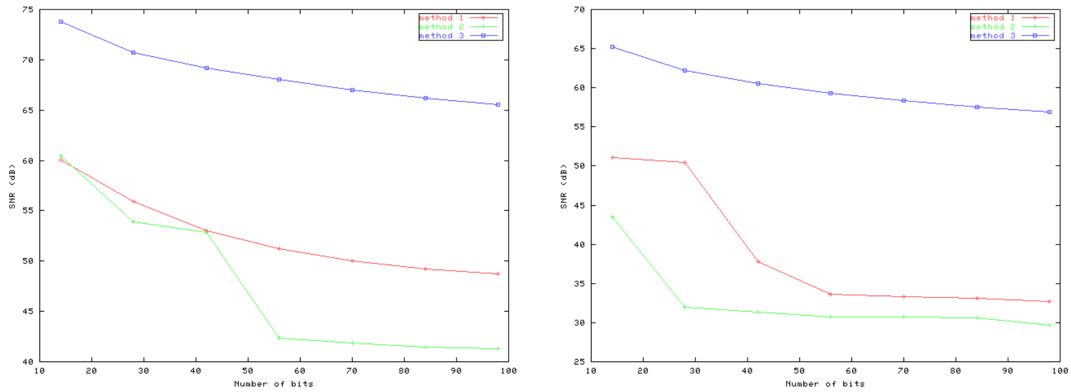
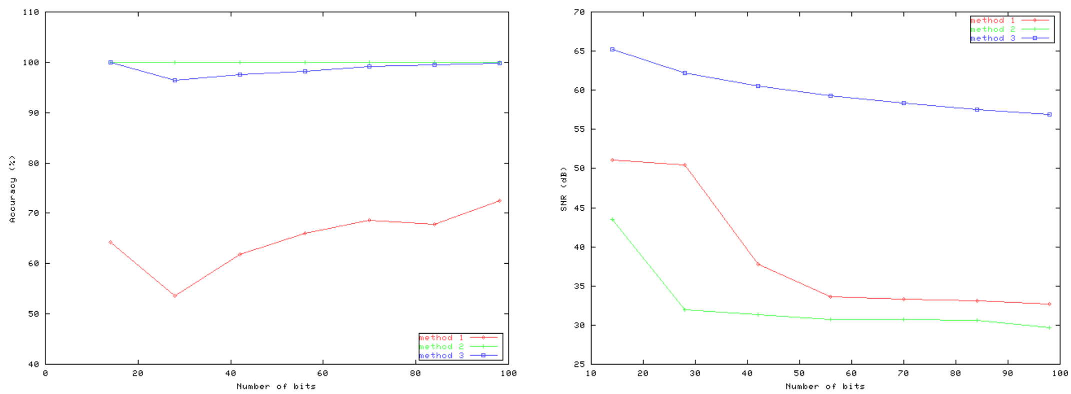


Figure 5.2: (a) SNR verses range block size . (b) Accuracy verses range block size . (c) Hiding time verses range block size . (d) Extracting time verses range block size. These results are for 256×256 Lenna image with data length 49 bits and range block sizes 2×2, 4×4, 8×8 and 16×16.



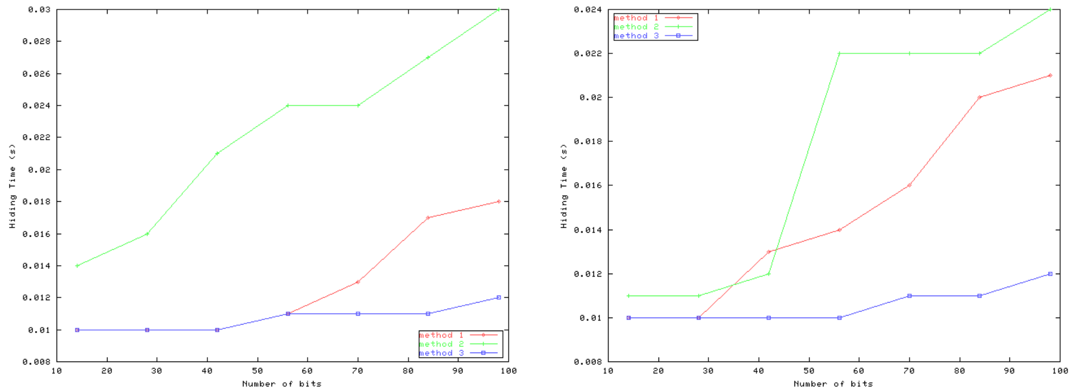
(a) (b)

Figure 5.3: (a) SNR verses number of bits hidden for 256×256 Lenna image with range block size 4×4. (b) SNR verses number of bits hidden for 256×256 Lenna image with range block size 8×8.



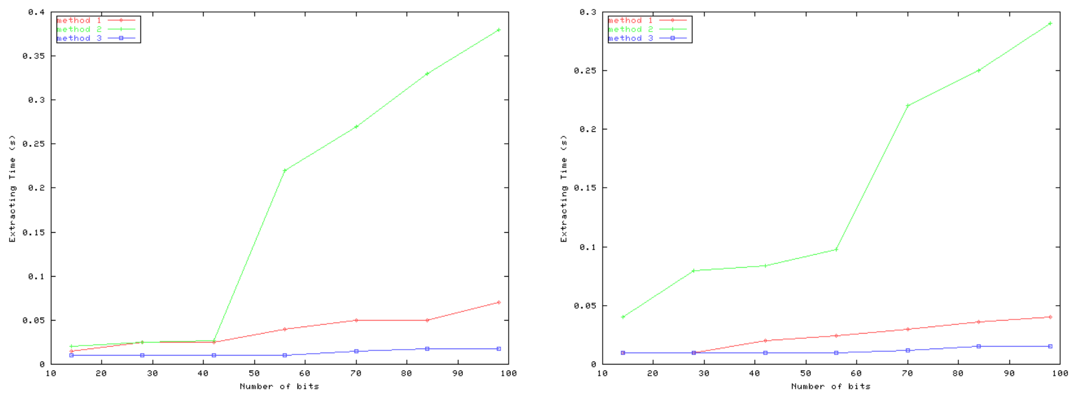
(a) (b)

Figure 5.4: (a) Accuracy verses number of bits hidden for 256×256 Lenna image with range block size 4×4. (b) Accuracy verses number of bits hidden for 256×256 Lenna image with range block size 8×8.



(a) (b)

Figure 5.5: (a) Hiding time verses number of bits hidden for 256×256 Lenna image with range block size 4×4. (b) Hiding time verses number of bits hidden for 256×256 Lenna image with range block size 8×8.



(a) (b)

Figure 5.6: (a) Extracting time verses number of bits hidden for 256×256 Lenna image with range block size 4×4. (b) Extracting time verses number of bits hidden for 256×256 Lenna image with range block size 8×8.

Brightness	Contrast	SNR	Accuracy	SNR	Accuracy	SNR	Accuracy
		Range block 4×4		Range block 8×8		Range block 16×16	
10	10	28.05	97.95	28.06	93.87	28.06	100.0
20	20	21.46	97.95	21.46	100.0	21.46	100.0
30	30	17.32	97.95	17.32	93.87	17.32	100.0
40	40	14.32	93.87	14.32	93.87	14.32	100.0
50	50	11.76	91.83	11.76	89.79	11.76	100.0

Table 5.1: Results for method 2 for different values of brightness and contrast.

Brightness	Contrast	SNR	Accuracy	SNR	Accuracy	SNR	Accuracy
		Range block 4×4		Range block 8×8		Range block 16×16	
10	10	28.05	63.20	28.06	75.51	28.06	81.63
20	20	21.46	65.30	21.46	57.14	21.46	63.26
30	30	17.32	59.18	17.32	69.38	17.32	67.34
40	40	14.31	44.89	14.31	44.89	14.31	57.31
50	50	11.76	57.14	11.76	67.34	04.77	38.77

Table 5.2: Results for method 3 for different values of brightness and contrast.

5.5.2 Addition of Noise

We checked the robustness of watermark by adding different amount of noise into the image which was watermarked by methods 2 and 3. Tables 5.3 and 5.4 show the results. Watermarked put by method 2 with range block size 16×16 was robust to some extent. 70 to 77 percent of the watermark could be retrieved when the noise was between 0.01 and 1.05. For smaller blocks watermark was not robust against the addition of noise.

Watermarked put by method 3 could not withstand against the addition of noise. Accuracy in retrieving the watermark was 40 to 55 percent only.

Noise	SNR	Accuracy	SNR	Accuracy	SNR	Accuracy
	Range block 4×4		Range block 8×8		Range block 16×16	
0.01	43.05	53.24	43.04	59.18	43.05	71.42
0.02	35.69	55.35	35.71	59.18	35.68	69.38
0.03	31.72	59.18	31.72	65.30	31.72	77.55
0.04	28.99	44.79	29.00	55.10	28.99	71.42
0.05	26.93	57.54	26.93	65.30	26.92	69.38

Table 5.3: Results for method 2 for different amount of noise.

Noise	SNR	Accuracy	SNR	Accuracy	SNR	Accuracy
	Range block 4×4		Range block 8×8		Range block 16×16	
0.01	43.06	53.30	43.05	53.30	43.06	59.18
0.02	35.74	45.40	35.75	53.30	35.74	55.10
0.03	31.81	49.88	31.81	52.68	31.81	57.14
0.04	29.01	42.46	29.00	44.89	29.01	53.06
0.05	26.92	47.14	26.92	47.14	26.92	53.06

Table 5.4: Results for method 3 for different amount of noise.

5.5.3 JPEG Compression

Methods 2 and 3 were tested against the JPEG compression with varying quality parameter. Method 2 shown satisfactory results against JPEG compression. About 92 percent of the watermark was correctly retrieved when hidden with the range block size 16×16 and quality parameter 1.0. For other parameters, accuracy was between above 70 percent. But for range block size 4×4 and lower quality values watermark is week (see table 5.5).

Quality	SNR	Accuracy	SNR	Accuracy	SNR	Accuracy
	Range block 4×4		Range block 8×8		Range block 16×16	
1.00	52.81	89.79	52.82	85.71	52.87	91.83
0.99	48.77	73.46	48.83	79.59	48.94	89.79
0.98	44.12	67.34	44.18	71.42	44.44	95.91
0.97	41.05	59.18	41.13	71.42	41.48	75.51
0.96	37.45	51.02	37.55	75.51	37.95	81.63

Table 5.5: Results for method 2 for JPEG compression with different quality parameter.

Quality	SNR	Accuracy	SNR	Accuracy	SNR	Accuracy
	Range block 4×4		Range block 8×8		Range block 16×16	
1.00	52.81	61.22	52.81	100.0	52.80	100.0
0.99	48.78	53.06	48.78	65.30	48.76	100.0
0.98	44.11	38.77	44.11	57.14	44.10	51.02
0.97	41.04	28.57	41.04	38.77	41.02	53.06
0.96	37.44	42.85	37.44	55.10	37.42	51.02

Table 5.6: Results for method 3 for JPEG compression with different quality parameter.

It was observed that the watermark put by method 3 with range block size 16×16 and quality parameter 1.0 and 0.99, was retrieved with 100 percent accuracy. Same result is obtained with range block size 8×8 and quality parameter 1.0. But for less quality parameter, watermark is very week. Also for range block size 4×4 watermark is not retrieved successfully (see table 5.6).

5.5.4 Scaling

Image was scaled after putting a watermark in it. Method 2 did not work well. The maximum accuracy obtained is 65 percent when the image was watermarked with range block size 8×8 and then image was scaled to its two times (see table 5.7).

Range Block Size	Scaling Factor	Accuracy
4×4	2	57.14
4×4	4	42.85
8×8	2	65.30
16×16	2	55.10

Table 5.7: Results for method 2 on scaling.

Range Block Size	Scaling Factor	Accuracy
4×4	2	26.53
4×4	4	75.51
8×8	2	100.0
16×16	2	100.0

Table 5.8: Results for method 3 on scaling.

Method 3 has shown some encouraging results. 100 percent accuracy obtained when the image was watermarked with range block size 8×8 and 16×16 and scaled to its double (see table 5.8). But for range block 4×4 with scaling factor 2, accuracy is poor.

5.6 Comparison

Several types of tests were carried out and it was seen that a single method is not best under all conditions.

SNR was calculated and method 3 came up with the highest value while the method 2 with lowest at all range block sizes. So, there is less distortion in image after putting data into the image, by method 3 (see figure 5.2(a)). Same results were seen in SNR verses number of bits. Here, SNR is very low in case of method

2 as compared to method 3 (see figure 5.3). It is clear from figures 5.2(b) and 5.4 that accuracy of method 1 is poor. Methods 2 and 3, for range block size larger than 2×2, shown good results in terms of accuracy in retrieved the data back. Method 3 takes the least time to hide and extract the information from the image for all block size (see figures 5.2(c) and 5.2(d)) and data length (see figures 5.5 and 5.6). Method 2 takes quite larger time for larger data length as compared to other methods.

On changing the brightness and contrast of the watermarked image, method 2 shown better results than method 3. On adding noise, method 2 works slightly better than method 3. Watermark put by method 3 is more robust than that of method 2 against JPEG compression. Also, on scaling the watermarked image, method 3 gave better results.

Chapter 6

Conclusion and Further Research

6.1 Conclusion

The number of bits of data that can be stored, depends upon the number of range blocks that have match in domain region. Bigger the range region more is the data that can be stored. But, since range region can not overlap the domain region, on increasing the range region, domain region is reduced which may lead to worse quality of image. So, there is a trade-off between the amount of data and quality of image produced.

Increase in tolerance level would allow to use all range blocks so that more data can be stored. However low tolerance is desirable in order to give an image that is visually close to the original.

6.2 Further Research

Further research should go towards improving the watermarking program and adding extra functionality. One of these is looking at having multiple watermarks for a single image, so that different parts of the image have a different watermark. There is also the need to further develop the robustness of existing watermarking techniques to combat the ever-increasing attacks on watermarks.

We used fixed partitioning scheme. Instead of this, adaptive partitioning scheme can be used, which would yield better results if used as the basis for the data hiding method.

In this thesis, various methods of steganography and fractal encoding have been explored but they have their own limitations. There is a big scope to explore the fractal characteristics in the wavelets. Wavelets can provide high level of performance in terms of quality and time.

Bibliography

[1] Fabien A.P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Attacks on copyright marking systems. In *Information Hiding*, pages 218–238, 1998. 8, 9

[2] Neil F. Johnson and Sushil Jajodia. Exploring steganography: Seeing the unseen. *IEEE Computer*, 31(2):26–34, 1998. 10

[3] A. Z. Tirkel R. G. van Schyndel and C. F. Osborn. A two-dimensional digital watermark. *Int. Conf. on Image Processing*, Vol. 2:86–90, 1994. 10

[4] E. Koch and J. Zhao. Towards robust and hidden image copyright labeling. In *Proc. of 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, pages 452–455, Halkidiki, Greece, June 1995. 11

[5] Ingemar Cox, Joe Kilian, Tom Leighton, and Talal Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, 6(12):1673–1687, 1997. 11

[6] D.Gruhl W. Bender and N. Morimoto. Techniques for data hiding. *IBM System Journal*, Vol.35, 1996. 11

[7] A. Jacquin. *A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding*. PhD thesis, Georgia Institute of Technology, August 1995. 15

[8] G. M. Hunter and K. Steiglitz. Operations on images using quad trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence 1, 2*, pages 145–153, 1979. 18

[9] Yuval Fisher. *Fractal Image Compression: theory and application*. Springer-Verlag, 1995. 20

[10] Narasimha Reddy Rachumallu. Visual cryptography - a fractal approach. 25

[11] J-M. Chassery P. Bas and B. Macq. Robust watermarking based on the warping of pre-defined triangular patterns. *In Proc. SPIE*, pages 99–109, 2000. 25

[12] J.-M. Chassery P. Bas and F. Davoine. A geometrical and frequential watermarking scheme using similarities. *SPIE Conference on Security and Watermarking of Multimedia Contents,San Jose*, 3657:264–272, 1999. 25

[13] P. Davern and M. Scott. Fractal based image steganography. *Information Hiding, First International Workshop, Lecture Notes in C omputer Science*, pages 279–294, 1996. 25

[14] J. Puate and F. Jordan. Using fractal compression scheme to embed a digital signature into an image. In *Proceedings of SPIE Photonics East'96 Symposium*, Boston, Massachusetts, 1996. 25

[15] Patrick Bas Jean-Marc. Using the fractal code to watermark images. 25

[16] J-M. Chassery P. Bas and F. Davoine. Self-similarity based image watermarking. *IX European Signal Processing Conference, Island of Rhodes, Greece*, pages 8–11, 1998. 25