Compression of the Mammography Images using Quadtree based Energy and Pattern Blocks

Spideh Nahavandi Gargari
M.S., Electronics Engineering, IŞIK UNIVERSITY, 2017

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Electronics Engineering

IŞIK UNIVERSITY
2017

IŞIK UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

Compression of the Mamoography Images using Quadtree Based Energy and Pattern Blocks

Sepideh Nahavandi Gargari

APPROVED BY:

Associate Professor Umit GUZ Işık University
(Thesis Supervisor)

Associate Professor Hakan Gurkan Işık University
(Thesis Co-advisor)

Professor Serdar Ozoguz Istanbul Technical University
(Jury Member)

Assistant Professor Sakip Onder Işık University
(Jury Member)

APPROVAL DATE: 01/16/2017

Compression of the Mammography Images using Quadtree Based Energy and Pattern Blocks

Abstract

Medical images, like any other digital data, require compression in order to reduce disk space needed for storage and time needed for transmission. This thesis offers , a novel image compression method based on generation of the so-called classified energy and pattern blocks (CEPB) is introduced and evaluation results are presented. The CEPB is constructed using the training images and then located at both the transmitter and receiver sides of the communication system. Then the energy and pattern blocks of input images to be reconstructed are determined by the same way in the construction of the CEPB. This process is also associated with a matching procedure to determine the index numbers of the classified energy and pattern blocks in the CEPB which best represents (matches) the energy and pattern blocks of the input images. Encoding parameters are block scaling coefficient and index numbers of energy and pattern blocks determined for each block of the input images. These parameters are sent from the transmitter part to the receiver part and the classified energy and pattern blocks associated with the index numbers are pulled from the CEPB. Moreover, in the second part of our method we used Quadtree too. By this way, all CEPB from quadtree results determined for each block of the input images too. input image is reconstructed block by block in the receiver part using a mathematical model that is proposed by 2 different method:

- Reconstruct Based on one block size
- Reconstruct Based on Quadtree

Evaluation results show that the method provides considerable image compression ratios and image quality even at low bit rates. Test result have shown that Compression ratio and PSNR results is acceptable, moreover, Quadtree method gives better results that fix based block size.

Quadtree Tabanlı Enerji ve Desen Bloklarını Kullanarak Mamografi Görüntülerinin Sıkıştırılması

Özet

Tıp görüntüleri, diğer dijital veriler gibi, depolama için gerekli disk alanını ve iletim için gerekli zamanı azaltmak için sıkıştırma gerektirir. Bu tez, sınıflandırılmış enerji ve desen blokları (CEPB) olarak adlandırılan yeni bir görüntü sıkıştırma yöntemi Tanıtılır ve değerlendirme sonuçları sunar. CEPB, eğitim görüntülerini kullanarak inşa edilmiş ve daha sonra ıletişim sisteminin verici ve alıcı taraflarnda kullanilir. Daha sonra giriş görü ntülerinin enerji ve desen blokları yeniden yapılanmasinda, CEPB'nin yapımı gibi aynı şekilde belirlenir.Bu işlem aynı zamanda girdi görüntülerinin enerji ve desen bloklarını da buluyor. Kodlama parametreleri giriş görüntülerinin her bloğu için, blok ölçeklendirme katsayısı, enerji endeksi sayıları, ve kalıp bloklarıdır. Bu parametreler, verici bölümünden alıcının parçası ve endeks numaraları ile ilişkili sınıflandırılmış enerji ve desen blokları CEPB'den çekilir. Dahası, yöntemimizin ikinci bölümünde Quadtree'yi de kullandık. Bu yöntem ile, Quadtree metodina bağlı, girdi görüntülerinin her bloğu için CEPB'leri de belirlendi. Giriş görüntüsü, 2 farkl yöntem tarafından önerilen matematiksel bir model kullanarak alıcı parça içerisinde blok blok olarak yeniden oluşturulmuştur:

- Sabit blok boyuna göre,
- Quadtree metoduna göre.

Değerlendirme sonuçları yöntemin düşük bit hızlarında bile önemli görüntü sıkıştırma oranları ve görüntü kalitesi sağladığını göstermektedir. Test sonucu, sıkıştırma oranı ve PSNR sonuçlarının kabul edilebilir olduğunu, ayrıca Quadtree yönteminin, blok boyutunu sabitleyen daha iyi sonuçlar verdiğini göstermiştir.

Acknowledgements

The author would like to thank to Professor Umit Guz, Hakan Gurkan (Isik University, Engineering Faculty, Department of Electronics Engineering), Assistant Professor Murat Gezer (Istanbul University, Engineering Faculty, Department of Electronics Engineering), the many helpful discussions on this work during this thesis. The author also would like to thank the anonymous reviewers for their valuable comments and suggestions which substantially improved the quality of this paper.

I dedicate this thesis to my parents Fatemeh Mehdinejad and Yadollah Nahavndi. I hope that this achievement will complete the dream that you had for me all those many years ago when you choose to give me the best education you could.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

More and more fields of humans life are becoming computerized nowadays. This determines generation of huge, and further increasing, amount information stored in digital form.

All this is possible thanks to technological progress in registration of different kinds of data. This progress is also being observed in wide field of digital images, which covers scanned documents, drawings, images from digital or video cameras, satellite images, medical images, works of computer graphics and many more.

Many disciplines, like medicine, e-commerce, e-learning or multimedia, are bounded with ceaseless interchange of digital images. A live on-line transmission of a sport event, or a surgery with a remote participation of one or more specialist, teleconference in a world wide company constitute great examples. Such utilization of technology related to digital images becomes nowadays very popular. Long-lasting storage of any data often can be very profitable.

In medicine, Hospital Information Systems contain a large number of medical examination results. Thanks to them doctors can familiarize themselves with the case history and make a diagnosis based on many different examination results. Such systems are also very useful for the patients because they gain access to their medical data. A very good example is IZIP Czech system, which gives Internet access to patients health records. Unfortunately, these hospital databases are growing rapidly each day tens or hundreds of images are produced and most of them, or even all, are archived for some period.

Both mentioned aspects of digital data sharing and storage, are linked with problems that restrain the progress in new technologies and growth of their application prevalence. During exchanging image data, one wishes to keep the quality on a high level and the time needed for transmission and the disk space needed for storage as low as it can be. The increase of throughput in used communication connections unfortunately is insufficient and some additional solution must be introduced to satisfy ascending expectations and needs. Collecting any kind of data results in demand for increase of storage devices capacity. Since the capacity growth of such devices is quite fast, almost any demand can be technically satisfied. However with extending of the capacity expenses, which cannot be passed over, are related.

Above-mentioned problems resulted in research and development of data compression techniques for Mammography images. With time many different compression methods, algorithms and file formats were developed. In still images compression there are many different approaches and each one of them produces many compression methods. However all techniques prove to be useful only in a limited usage area. Of course, image compression methods are also much desired or even necessary in medicine. However, medical images require special treatment because correctness of diagnosis depends on it specially in Mammography images that details are very small and important in order to define the problem.

Low quality medical image, distortions in the image or untrue details may be harmful for human health. Thus any processing of such images, including compression, should not interfere in the information carried by the images.

1.2 Research Aim and Objectives

Raw or uncompressed multimedia data such as graphics, still images, audio, and video requires huge storage capacity and transmission bandwidth. It would be needed to find efficient ways to encode the audio signals and images. Moreover, high compression ratio and fast communication technology required. There are uniform or plain areas in image that contain adjacent picture elements (pixels) that have the same numeric values .By this way it would be large number of spatial redundancy (or correlation between pixel values that are numerically close to each other.

To remove this redundancy in order to get more efficient ways to represent the still images the compression is needed. The performance of the compression algorithm is measured by the compression ratio (CR) and it is defined as a ratio between the original image data size and compressed image data size.

In general, the compression algorithms can be grouped as *lossy and lossless* compression algorithms.

In the lossy compression method, the image compression algorithm should achieve a trade off between the image quality and the compression ratio. It should be noted that, higher compression ratios produce lower image quality. Moreover, the image quality can be effected by the other characteristics, some details or content of the input image.

The compression performance of these methods is affected by several factors such as block size, entropy, quantization error, truncation error and coding gain.

Based on the results of experiment have been done by transforming two-dimensional images from the spatial domain to the frequency domain, It has been proved that, the human visual system (HVS) is more sensitive to energy with low spatial frequency than with high spatial frequency. While the low spatial frequency components correspond to important image features, the high frequency ones correspond to image details. Therefore,in order to compression it would be needed to quantization and transmission the most important or low-frequency coefficients while the remaining coefficients are discarded.

In order to achieve this goal in compression, The uniformly sized image blocks used to reach the compression. In this method, it does not take into account the irregular regions within the real images. The fundamental limitation of the DCT-based compression is the block-based segmentation or framing. In these methods, depend on the block size of the images, the degradation which is also known as the blocking effect occurs. A larger block leads to more efficient coding or compression but requires more computational power. Although image degradation is noticeable especially when large DCT blocks are used, the compression ratio is higher. Therefore, most existing systems use image blocks of 8×8 or 16×16 pixels as a compromise between coding or compression efficiency and image quality. In this paper, a new block-based image compression scheme is proposed based on generation of fixed block sets called Classified Energy Blocks (CEBs) and Classified Pattern Blocks (CPBs). All these unique block sets are associated under the framework called Classified Energy and Pattern Blocks (CEPBs). Basically, the method contains three main stages:

1. Generation of the CEPB,
2. Encoding process which contains construction of the energy and pattern building blocks of the image to be reconstructed and obtaining the encoding parameters.In this step 2 different method has been done:
   - Find encoding parameter based on *fixed block size*,
   - Find encoding parameter based on *Quadtree*.
3. Decoding (reconstruction) process of the input image using the encoding parameters from the already located CEPB in the receiver part (decoding).

In this thesis, the size of the image block vectors (LIBV)is set to $L_{IBV} = i \times j, i = j = 2, 4, 8$ to construct the CEPB. It is observed that, when the compression ratio reaches the higher levels, degradation in the image caused by the blocking effect is getting visible.

The speed of the algorithm and the compression ratio are also increased by adjusting the size of the CEPB with an efficient clustering algorithm in both group of experiments. Moreover, as In medical images as Mammography image there are very similarity in details, it was prefered to use Quadtree method too.

1.3 Research Questions

The thesis addresses the following research questions:

- Research Question 1 : Is it possible, and how to minimize the drawbacks of Classified Energy and Pattern Building Block (CEPB) method on Mammography images compression?
- Research Question 2: How can Quadtree method affect the (CEPB) method results on Mammagraphy images in comparison by (CEPB) method based on fixed block size?
- Research Question 3: Does the (CEPB) method preserve image quality and how affects the image quality ?

1.4 Thesis Outline

- *Chapter 1* consists of introduction part,
- *Chapter 2* explains basic concepts of digital imaging and discusses characteristics of imaging.
- *Chapter 3* explains details of Medical images and history of it specially about Mammography images that has been used in thesis.
- *Chapter 4* provides basic information about image compression and fractal image compression, what is preceded with general description of image compression, the different method of compression and fractional compression.
- *Chapter 5* goes into details about clustering and K-means clustering that have been used in thesis method.
- *Chapter 6* gives a look inside the implemented algorithm.
- *Chapter 7* presents and discusses the results of experiments that were performed on the implementation of the proposed fractal compression method.
- *Conclusion* presents the discussion of the results and recommendations.The answers to the research questions can be found here.

Chapter 2

Digital Imaging

Before medical imaging and Mammography Images will be discussed, it is necessary to provide some basic information about digital imaging. The digital images are described in the first section and the next section concentrates on a specific class of digital image characteristic.

2.1 Analog and Digital Images

Two classes of images can be distinguished, analog and digital images. Both types fall into non temporal multimedia type.

Analog images are painted or created through photographic process. During this process, the image is captured by a camera on a film that becomes a negative. We have a positive when the film is developed no processing is possible from this moment. When the photography is made on a transparent medium then we are dealing with a diapositive (a positive photographic slide or transparency). Analog images are characterized by continuous, smooth transition of tones. This means that between each two different points at the picture there is an infinite number of tonal values. It is possible to transform an analog image into digital. The digitization process is usually caused by a need of digital processing. The output of digitalization is a digital approximation of the input analog image the analog image is replaced by a set of pixels (points organized in rows and columns) and every pixel has a fixed, discrete tone value. Therefore, the image is not a continuous tone of colors. The precision and accuracy of this transformation depends on the size of a pixel the larger area of an analog image transformed into one pixel the less precise approximation [1].

Digital image can be captured with a digital camera, scanner or created with a graphic program. Transition from digital to analog image also takes place by such devices as computer monitor, projector or printing device. One can distinguish many different types of digital images. First of all the digital images are divided into recorded and synthesized images. To the first group, for example, belong analog images scanned by digital scanner. To the second group are classed all images created with graphical computer programs they come into being already as digital images.

The second possible classification of digital images divides them into vector images and raster images. Both of the groups can contain recorded as well as synthesized images. Vector images mostly are created with graphic software. Analog images can be recorded only to a raster image, but then they can be converted to vector image. The opposite conversion (rasterisation) is also possible. Vector images are treated as a set of mathematically described shapes and most often are used in creating drawings like logos, cartoons or technical drawings. This work concerns only raster graphics, where an image (bitmap) is defined as set of pixels (picture elements) filled with color identified by a single discrete value. This kind of images is usually used for photographical images [2].

2.2 Digital Image Characteristics

Digital images are characterized by multiple parameters. The first feature of a digital image is its color mode. A digital image can have one of three modes:

- Binary,

- Gray scale,
- Color.

A binary (Bi-level) image is an image in which only two possible values for each pixel. A grayscale image means that its each pixel can contain only a tint of gray color. As it was already mentioned, a digital image is a set of pixels. Each pixel has a value that defines color of the pixel. All the pixels are composed into one array.

The resolution of a digital image is the number of pixel within a unit of measure [3]. Typically,the resolution is measured in pixels per inch (PPI). The higher image resolution the better is its quality. The image resolution can also be understood as dimension of the pixel array specified with two integers [2]:

$$\textit{number of pixel columns} \times \textit{number of pixel rows} \tag{2.1}$$

Bit depth, called also color depth or pixel depth, stands for how many bits are destined for description of color for each pixel. Higher color depth means that more colors are available in the image, but at the same time, it means that more disk space is needed for storage of the image. Monochrome images use only one bit per pixel, and gray scale images engage usually 8 bits, which gives 256 gray levels. Color images can have pixel depth equal 4, 8 or 16 bits; full color can be achieved with 24 or 32 bits. Colors can be described in various ways. Next digital images feature color model not only specifies how colors are represented, but also determines the spectrum of possible colors of pixels. The gamut of colors that can be displayed or printed depends on color model that is employed. This is why a digital image in a particular color model can use only a portion of visible spectrum this portion is characteristic for the model.

There are many different color models and most popular are: RGB, CMY, CMYK, HSB (HSV), HLS, YUV, and YIQ. These color models are divided into two classes: subtractive and additive. CMY (Cyan, Magenta, and Yellow) and CMYK (Cyan, Magenta, Yellow, and Black) are subtractive models. One should make use of one of model from this class when printing inks (with presence of external light that is being reflected by printed image) must be employed to display color. RGB (Red, Green, Blue), one of additive models, is used when displaying color with emission of light e.g. image is displayed by computer display monitor. The main difference between these two kinds of color models is that in subtractive models black is achieved by combining colors and in additive models in this way is produced white. In subtractive models, colors are displayed thanks to the light absorbed (subtracted) by inks. While in additive models, colors are displayed thanks to the transmitted (added) light. HSB (Hue, Saturation, and Brightness), also called HSV (Hue, Saturation, and Value), is more intuitive color of a pixel is specified by three values:

- **Hue**: the wavelength of light,
- **Saturation**: the amount of white in the color,
- **Brightness**: the intensity of color.

Similar to HSB and also very intuitive is HLS (Hue, Lightness, and Saturation). The YUV color model is a part of PAL system in television and contains three components one for luminance and two for chrominance. YIQ also has one component for luminance and two components for chrominance and it is used in NTSC television.

Channels are closely related with color models. A channel is a grayscale image that reflects one of color model component (base of color in used color mode). Channels have same size as the original image. Thus:

- an image in RGB will have 3 channels: Red color, Green color, Blue color.
- in CMYK four channels: Cyan color, Magenta color, Yellow color, Black color.

- HSV will have three channels: Hue value, Saturation value, Brightness value.
- a grayscale image will have only one channel.

There can be additional channels called Alpha Channels. An Alpha channel stores information about transparency of pixels. Therefore, the number of channels, although it partially depends on the color model, is also a feature of a digital image. Colors indexing is next image feature related to color model. Indexed color model is only an option and means that number of colors that can be used is limited to a fixed number (e.g. in GIF to 256).

In order to reduce the bit depth and size of whole file. Most often indexing is done automatically in accordance to standard palettes or system palettes. Palettes in different operating systems are not the same they only partially overlap. From 216 colors that are common for operating systems a standard palette was created for purpose of World Wide Web.

There are also other standard palettes a palette with 16 colors is commonly used for simple images. Besides indexing to standard/system palettes, there exists also adaptive indexing. In this indexing, the color space is reduced to a fixed number of colors that most accurately represent the image. Not necessarily all colors needed by the image must be indexed, but they can be. The difference between indexing to standard/system palette and adaptive indexing is that adaptive indexing requires definitions of colors from the palette at the beginning of the file and standard palettes do not have to be attached [3].

File format is next characteristic of a digital image. A digital image can be stored in one of many file formats. Some formats are bounded with one specific program, but there are also common formats that are being understood by different graphic programs.

There is a very close relation between file formats and compression. Images stored in a particular format are usually compressed in order to reduce the size of the file. Each format supports one or few compression methods; there are also formats that store uncompressed data [2].

The last characteristic of a digital image is compression method used to reduce size of file containing the image.

As a conclusion, there is more than one method to reduce amount of disk space needed to store a digital image. The most obvious one is compression, but there are also other, simpler like reduction of image resolution. There can be also decreased number of colors or introduced index to used color palette.

Chapter 3

Medical Images

Medical Imaging came into being in 1895 when W. K. Roentgen discovered X-rays. This invention was a great step forward for non-invasive diagnostics and rewarded with Nobel Prize in 1901. With time, other discoveries in the field of medical imaging were made that, like X-rays, support medicine and make possible more accurate and effective diagnosis. It is not feasible to list all of discoveries and inventions. Similar situation is with describing all types of medical images. Thus, only the most important discoveries will be mentioned with a characterization of images, which are products of these technologies. Although X-rays were discovered over a century ago, they are still in common use. During examination, the patient is being placed between an X-ray source and a detector. Different tissues absorb x-rays with different force thus the X-rays that went through the patient have different energy depending on what tissues they ran into. Dense tissues, e.g. bones, block and soft tissues give no resistance to the X-rays. Parts of the detector that are behind tissue that absorbs X-rays in 100% produce white areas on the image. The softer a tissue is the darker becomes the image in parts that represent this tissue.

Many different X-ray detectors can be used during medical examination. They can be divided into two classes. One class contains detectors, like photographic plate, that give analog images. These images can be transformed into digital image by process of digitalization.

The second class of detectors consists of devices that directly produce digital images. The most familiar detectors that fall into the second class are Photostimulable Phosphors (PSPs), Direct Semiconductor Detectors and combination of Scintillator with semiconductor detectors. In 1971 G. Hounsfield build first Computerized Tomograph (Computerized Axial Tomograph) an X-ray machine that produces a set of two-dimensional images (slices), which represent and three-dimensional object. For his invention, G. Hounsfield was awarded with Nobel price in 1979. Pictures created during tomography are called tomograms and they create a specific class of X-ray images.

There are also other classes, for example mammography images that has been used in this thesis. Apart from X-rays, also other technologies are used in medical imaging such as Gamma-ray imaging, radio waves (Magnetic resonance imaging (MRI) Images) etc. that were not described here.

Gamma-ray imaging is used in a field of nuclear medicine. In contrast to X-rays, here is no external source of gamma rays. A radioactive isotope, which emits gamma rays during decay, is administered to patient. Then the gamma radiation is measured with a gamma camera (gamma-ray detectors). Most popular applications of gamma rays in medical diagnosis are bone scan and positron emission tomography (PET). Bone scan with gamma rays can detect and locate pathologies like cancer or infections. PET generates a sequence of images that, like in X-ray tomography, represent a 3-D object.

Medical imaging employs also radio waves. Magnetic resonance imaging (MRI) is a technique in which short pulses of radio waves penetrate through a patient. Each such pulse entails response pulse of radio waves generated by all tissues. Different tissues emit a pulse with different strength. The strength and source of the each response pulse is calculated and a 2-D image is created from all of gathered information.

Ultrasound imaging in medical diagnostic composes ultrasonography. An ultrasound system consists of a source, a receiver of ultrasound, a display and

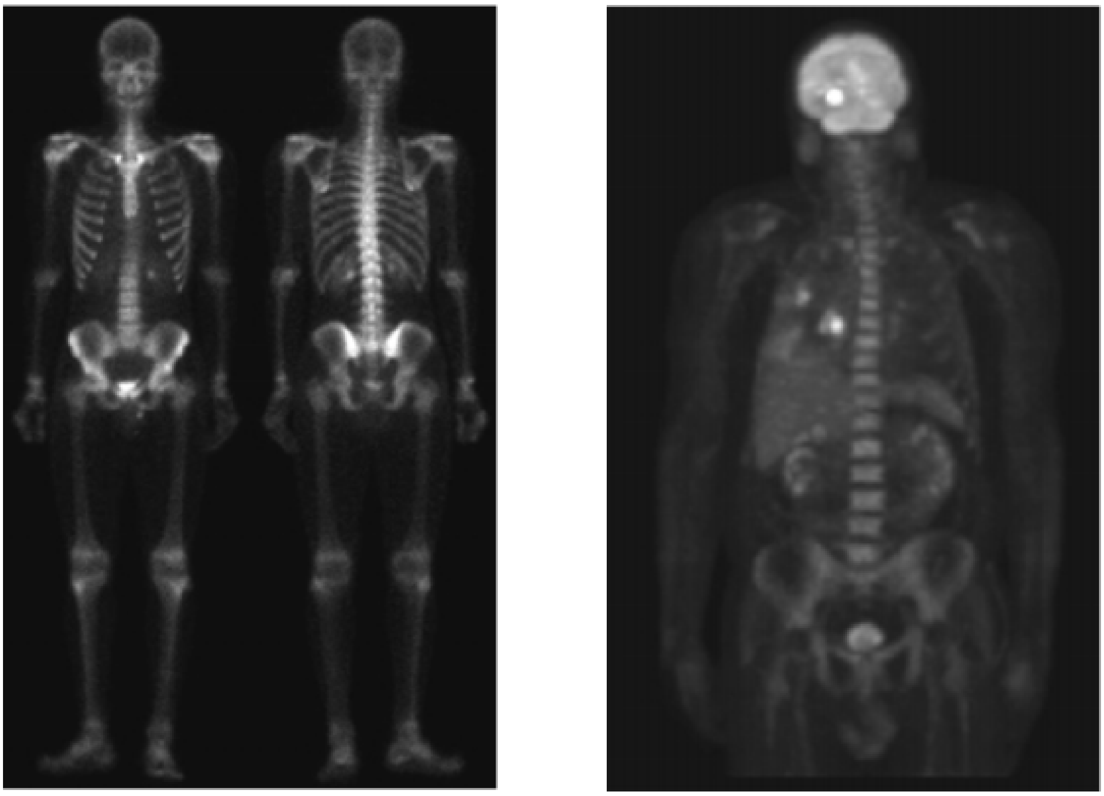


Figure 3.1: Examples of gamma-ray images [1]

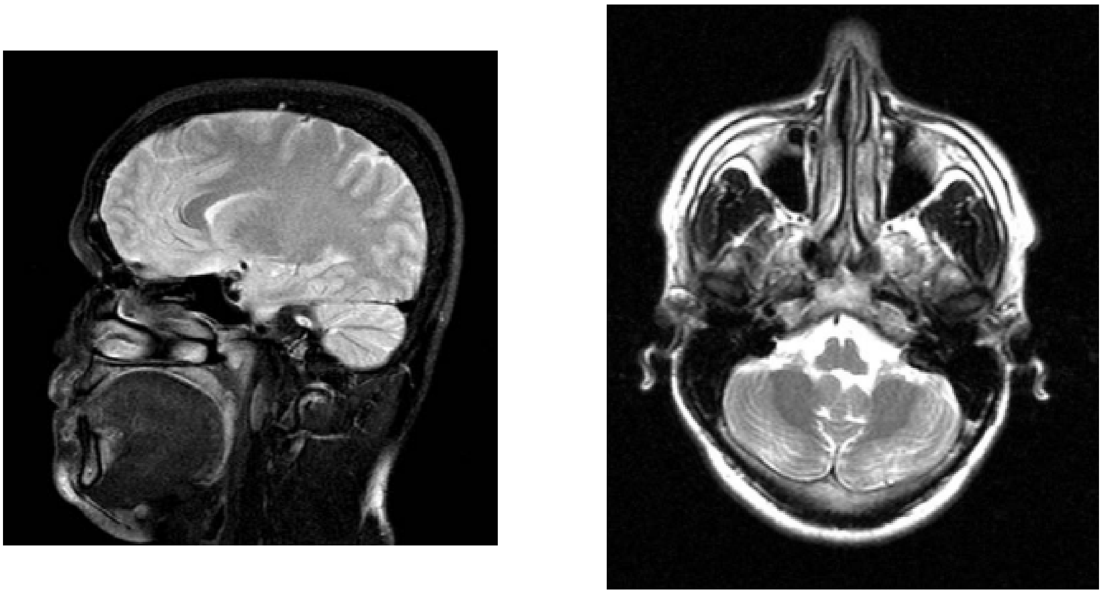


Figure 3.2: Examples of magnetic resonance images, (from normartmark, 21.09.2007).

a computer. High-frequency sound, from 1 to 5 MHz, is sent into the patient. Boundaries between tissues partially reflect the signal and partially allow it to pass. This means that the waves can be reflected on various depths. The ultrasound receiver detects each reflected signal and the computer calculates the distance between the receiver and the tissue, which boundary reflected the waves. Determined distances to tissues and strengths of reflected waves are presented on the display, i.e. they constitute a two-dimensional image. Such image typically contains information about millions of ultrasound signals and it is updated each second.

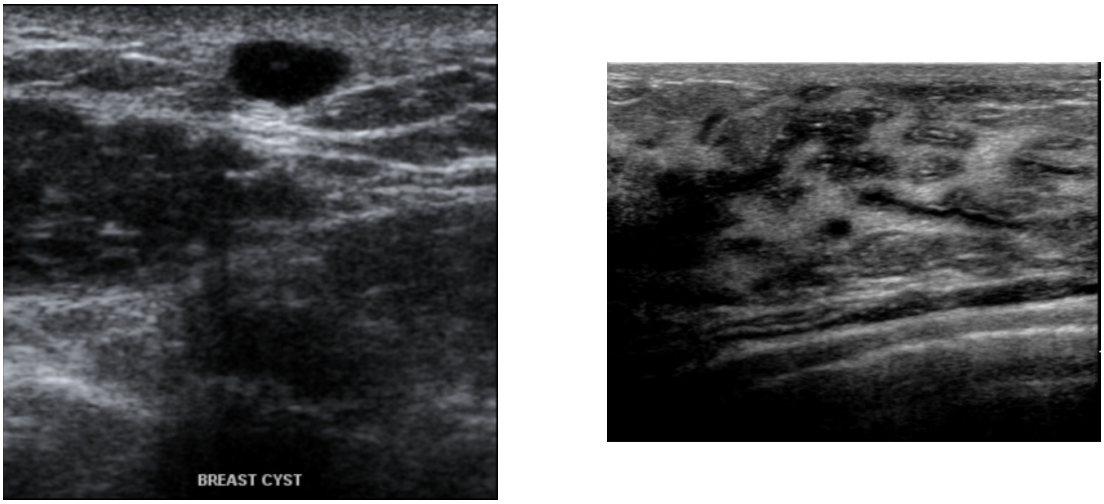


Figure 3.3: Examples of breast ultrasound images(from ultrasoundpaedia)

The review of medical imaging techniques unveil large diversity of medical images classes and technology used in medical diagnosis. Nevertheless, all these images have some common characteristics. All above-mentioned classes of medical images are characterized with very restricted size. Although there are color medical images, the most of them are monochromatic. Images from different classes have different sizes. Largest are the X-ray images, which can have size up to 2048 pixels vertically and horizontally. Other medical images are much smaller, for example Computerized Tomography are smaller than 512×512 pixels, Magnetic Resonance images up to 256×256 pixels and ultrasound images 700×500 or less [4].

3.1 Mammography Images

The important issue in order to decrease the breast cancer damages is the early detection before its symptoms.

Mammography remains the most valuable and successful technique for the early detection of breast cancer in breast imaging; a mammogram is a low dose an x-ray (radiography) picture or mammography exam of the breast. X-ray mammography is the only proven method detects non palpable cancers (Breast Cancer Detection, Lawrence).

In the 1960s, the first randomized controlled trial of screening with mammography was initiated in a health insurance program in New York, to test whether screening asymptomatic women for breast cancer could lower the death rate. The trial involved 62,000 women between 40 and 64 years of age. By comparing the subsequent number of deaths among the screened women with those in the control group, the investigators demonstrated that early detection could decrease the mortality from breast cancer. Now, most western countries have national programs to offer annual or bi-annual screenings to women above a certain age [5].

Studies have shown that the mammography exam is easier to women with fatty breasts than those with dense breasts (Breast Cancer). There are two types of mammograms:

- **Screening mammogram**, used to early detect breast cancer before its symptoms
- **Diagnostic mammogram**, used to evaluate patients with abnormal clinical findings and under treatments of breast cancer.

The quality of a mammogram image related on various items:

- **t**he nature and accessories of the mammography unit,

- the use photographic film for image acquisition,
- the storage and display,
- film resolutions, which the high resolution in film decrease signal contrast,

This limits the exposure dynamic range of mammography screen film systems to a factor of 2550, which means that the image contrast in the fully glandular or fully adipose parts of the breast can be much lower than in the other areas. On the other hand, digital detectors have the significant advantage of a linear response over a wide of exposure conditions, giving constant contrast and a large dynamic range. A mammography unit is in a shape of a box that have a tube that lodges x-rays used exclusively for breast x-ray exam.

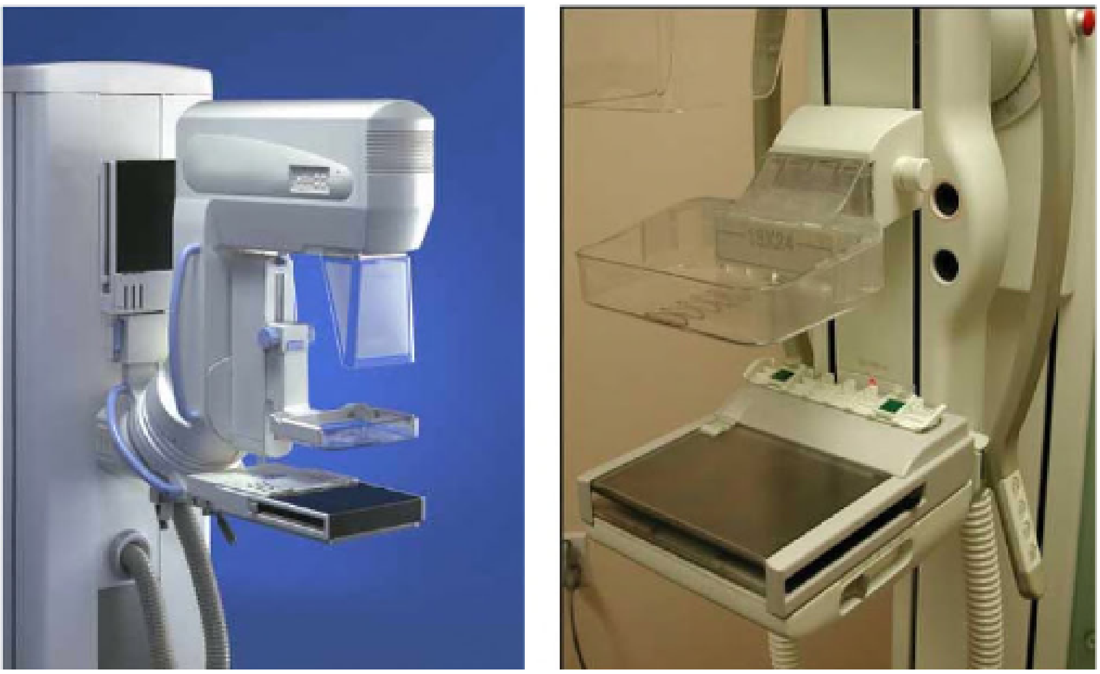


Figure 3.4: Mammography unit.

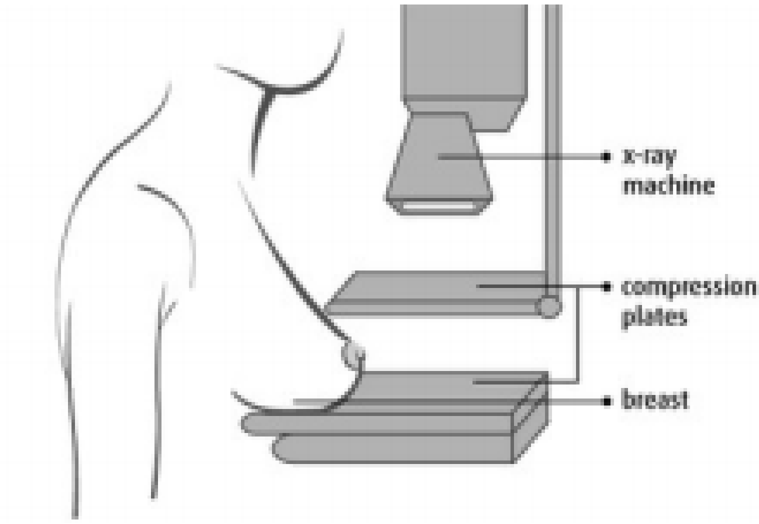


Figure 3.5: Mammography Screening [6].

There are lots of parameter that can affect mammography procedure quality as, the nature of the film, technology and skills used as well as the mammography knowledge, the accommodation of all patients ages, breast shapes and sizes.

A screen film mammography uses a low dose radiography (x-ray) film to acquire, store and display images. The transmitted x-rays are recorded in a film cassette; different parts of the body absorb and attenuate x-rays in varying degrees regarding the type of tissue. Fat organs pass x-rays whereas dense tissues absorb them due to their physical properties (Coiera). With the modern technology, the digital technology is taking over the analog technology. Analog refers to the assumption of an arbitrary value, but digital (or discrete) systems assume only few values. For the breast imaging, the transition to digital improves the quality of a mammography image; hence, the advancement of the early detection of breast cancer.

FFDM which is a new version of mammography unit is defined as a mammography system in which the x-ray film is replaced by solid-state detectors that convert x-rays into electrical signals. These detectors are similar to digital cameras' detectors. The electrical signals are used to produce images of the breast that can be seen on a computer screen or printed on special film similar to conventional mammograms (Bassett and Gold). Digital mammography improves the signal to noise ratio and the image contrast to enable the better detection of breast cancer (Daffner). 4 In order to gain best quality of images in dedicated mammog-

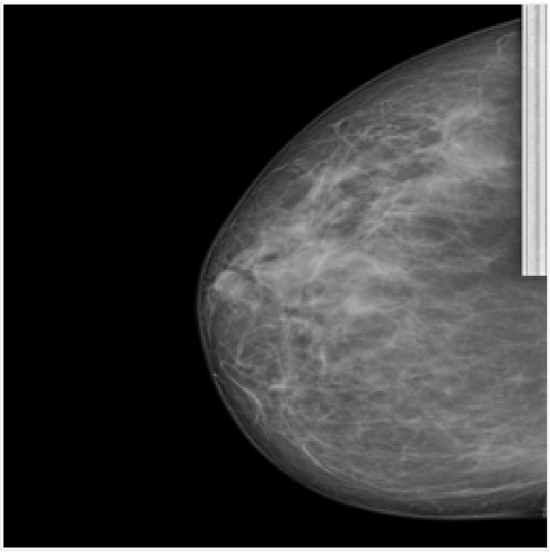


Figure 3.6: Mammography Image (from images used in thesis).

raphy unit, it is important to visualize different parts of the breast with sufficient details to deeply detect tissues from which cancer can develop. The equipment should be designed in a way that they can provide clinically useful information about images stored on the x-ray film and acquire all the necessary special accessories to improve the exposition of the breast tissues to x-rays; those factors include but not limited to: the choice of the film, processing technique, performance, electrical requirements, density selection, source-image detector distance, voltage, current and time selections to name few.

For mammography exam, a qualified radiology technologist positions the breast on a platform that is gradually compressed by a paddle to adjust a breast so that images can be taken at various angles; to avoid blurred images and change positions slightly between images, a patient might be asked to keep from breathing for few seconds. Once the part to be examined is carefully aimed, the x-ray machine passes a small burst of radiation through the body; therefore, images are recorded on a special cassette film for film screening mammography or directly digitally to a computer for digital mammography.

Once the exam is completed which takes about half an hour approximately (Lower time in new machines), a patient will be required to wait for more minutes for a technologist to check if images are of high quality for a doctor to read and interpret. The patient will be notified for the results after the doctor reads and interprets the obtained images.

3.1.1 Important Findings in Mammography

- Masses,
- Calcification,
- Architectural Distortion,
- Asymmetries (asymmetry, global asymmetry, focal asymmetry, developing asymmetry),
- Intramammary lymph nodes (IMLN),
- Skin lesion,
- Solitary dilated duct.

Among these abnormalities, Intramammary lymph nodes (IMLN), skin lesion and solitary dilated duct are rarely significant [7]

Mammography density is one of the most important issue in mammography imaging that refers to the prevalence (and to some degree the distribution) of fibroglandular tissue in the breast as it appears on a mammogram. The fibrous and glandular tissues cannot be distinguished in mammography due to a combination of physiological intertwining and similar x-ray attenuation coefficients. These tissues can, however, be distinguished from fatty tissue, which attenuates x-rays to a lower degree. This causes the fibroglandular tissue to stand out as bright areas on a dark background and therefore the term density is used to describe its appearance.

The first to correlate mammography density with risk of breast cancer was J N Wolfe [8], and his research in the mid 1970s lead to an early, four category, classification scheme now referred to as Wolfe patterns. This method is based on qualitative, visual assessment and contains the following four classes corresponding to ascending magnitude of risk.

In digital mammography, digital format let the easy application of digital image processing. In breast imaging where images are enhanced to detect and diagnose breast cancer lesions, the image enhancement bases upon the contrast enhancement to improve the image display; we can mentioned peripheral equalization in which the areas under the pectoral muscle and near the periphery of the breast are made brighter or darker to match the appearance of the tissue in the center of the breast.

This method lets the radiologists to review the whole breast without the manual adjustment to the viewing window or level. Another method is the image edge enhancement which consists of smoothing edges to making small objects more visible, such as calcification or speculated mass. In order to help radiologist to have better and easier detections **the computer-aided diagnosis systems** used, in which the image processing techniques are combined with artificial intelligence algorithms along with radiological image processing. Radiologists use CAD to read and interpret radiography images; the CAD system examines digitized film mammograms for evidence of suspicious masses or calcification. Radiologists can display the findings and match with the results by scanning images.

Chapter 4

Image Compression

This chapter written about image compression and medical image compression. The first two sections consist of general explanation medical compression and why we choose fractal compression for our thesis. The second section is explanation of fractal compression and details of different method that can used by fractal compression. In our method we choose Quadtree method that all details about is explained.

4.1 Fundamental of Image Compression

A compression method consists of processes: **compression and decompression**. In order to analyse Compression method we should discuss about two definition: compression, decompression,

Compression is used in order to represent original data by smaller number of bits.

Decompression is opposite process of compression that is used to reconstruct original data.

There is two types of compression method that will be distinguished in this part: Lossy Compression, and lossless Compression,

Lossless compression methods, reconstruction of data set during the decompression is similar as the original data set.

Lossy methods, reconstruction of data set is only approximate of the original data. so, the compression is immutable.

In order to have better efficiency of compression the similarity between original data and reconstructed data gets lower. To analyse visual difference between original and decompression images status of observation is highly effected. Moreover, image processes as image analysis show, if compression actually was not lossless, noise elimination will be defined. **"Visually lossless"** compression is lossy compression method when loss of information A lossy compression method is called **visually lossless** when the loss of information caused by compression-decompression is invisible for an observer. However, it related to the observer and conditions.

There are many factors to survey the efficiency of the compression, we will speak about.

- Compression ratio CR: the ability of the compression method to reduce the amount of disk space needed to store the data. CR is the most used parameter to evaluate efficiency of the compression.

  CR is defined as number of bits of the original image B_{org} per one bit of the compressed image B_{comp}:

  $$CR = \frac{(B_{org})}{B_{comp}}$$

  The compression percentage CP serves the same purpose:

  $$CP = (1 - \frac{1}{CR}).100\%$$

- Bit rate BR: The average number of bits in compressed representation of the data per element (symbol) in the original set of data represents Bit rate

BR . High effectiveness of a compression method manifests itself in high and CP , but in low BR.

- time needed for compression: to evaluate it different factors as product of time and bit rate should be used.

Here only the most commonly used factors were mentioned, but there are many more factors and methods to evaluate the efficiency of data compression.

4.1.1 Lossless Compression

Lossless image compression methods used mostly in compression methods. what lossless compression do is converting an input sequence of symbols into an output sequence of codewords.

One codeword usually corresponds to one element (symbol) in the original data; if we use stream coders, it will correspond to a sequence of symbols. The length of codewords can be fixed or variable.

In order to decompression, what we should follow is decoding of the code sequence.as we said before, The output of the decoding in Lossless compression is the same as the input of it . To use stream we should encoded it into parts by explicit bounded codeword.

Lossless compression method contains two phases: **modeling and coding**. The modeling phase used to build a model for the data to be encoded, that describes data information.

In order to choose true model method of the modeling for a specific compression technique should pay attention to a large extent on the type of compressed data, but it always focus on assessment of the input sequence, its regularities and similarities [9] .

Briefly, the model is a different, more ordinary representation of the original data that eliminates the redundancy [10]. The coding phase is based on a statistical analysis and strives after the shortest binary code for a sequence of symbols obtained from the modeling phase [10]. In this phase the analytical tools from information theory are commonly used.

Three groups are distinguished in lossless compression methods:

- entropy-coding,
- dictionary-based,
- prediction methods.

there are lots of compression techniques as various compression techniques can be found in a great number, Shannon-Fao coding, Huffman coding, Golomb coding, Unary coding, Truncated binary coding, Elias coding.

Within entropy coding methods also arithmetic methods can be found, e.g. In the first group entropy coding methods. Methods as Lempel-Ziv-Welch (LZW) coding, LZ77 and LZ78, Lempel-Ziv-Oberhumer algorithm, Lempel-Ziv-Markov algorithm can be mentioned for dictionary-based methods.About prediction methods we can mentioned JPEG-LS and lossless JPEG2000 algorithms that get popular now a days. With lossless compression is bounded a limitation that is shown by information and coding theory. The average length of codeword cannot be smaller than the entropy (expressed in bits) of the information source.

So the closer a compression technique comes to this limit the better compression ratio can be achieved, and no lossless compression method can come beyond this limit. The basic concepts of information theory are explained below.

Information is a term that actually has no precise mathematical definition in information theory. It should be understand in colloquial way and treated as indefinable. Information should not be confused with data (data build information) or message (transmitted information). Although there is no definition, it is possible to measure information. The amount of information is calculated thanks to following equation:

$$I(u_i) = \log \frac{1}{P_i}$$

where p_i is the probability that the symbol u_i will occur in the source of information. This equation measures the information related with occurrence of a single symbol in a probabilistic source of information. The unit of this information measure depends on the basics of the logarithm. When $bs = 2$ then the unit is bit, when $bs = 3$ then the unit is *bit*, when $bs = e$ (natural logarithm) then the unit is *nat*, and the last unit *Hartley* is used when $bs = 10$.

Entropy is a different measure of information it describes the amount of information specified by a stream of symbols. According to Shannon definition, the entropy is the average amount of information I_{u_i} for all symbols u_i that build the stream. So when data $U = u_1, u_2, ..., u_{\bar{U}}$ constitute the information then the entropy can be calculated from:

$$H(U) = \sum_{i=1}^{\bar{U}} p(u_i).log_{bs}\frac{1}{p(u_i)} = -\sum_{i=1}^{\bar{U}} p(u_i).log_{bs}p(u_i)$$

Above-mentioned formulas are correct only when emission of a symbol by the source is independent from past symbols i.e. when the source is memory less source. Other types of sources, e.g. source with memory or finite-state machine sources, like Markov source, require consideration of changes in these formulas.

4.1.2 Lossy Compression

Low compression ratios in lossless compression limit efficiency of this technique for compression that demand different approach to compression to make it better.

In order to have better efficiency, it needs to disposing of the reversible character of the encoding process.

In lossy compression methods it needs to reduce the information of the image to be encoded up to some level that is acceptable by a particular application field. So, in lossy methods there is a characteristic that evaluate the efficiency of compression distortion rate. distortion rate defines the distance between original image and the image reconstructed in decoding process. All other characteristics as compression ratio and time needed for encoding and decoding, etc. in lossless method are important and used too.

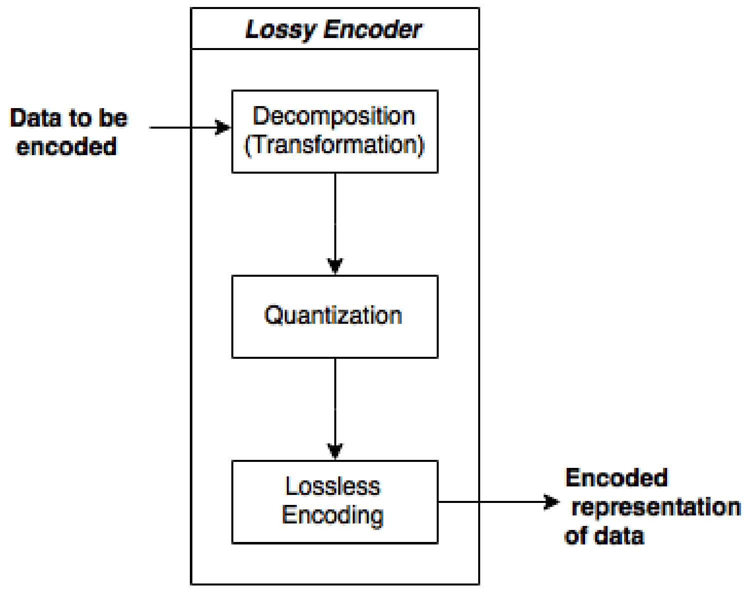


Figure 4.1: General scheme for lossy compression.

In lossy compression algorithms, it can be distinguished two obligatory phases. This means that the key issue for lossy methods is the quantization. Decomposition can be used too. it is optional, but very frequently used as it allows to have more efficiency.

Basic way to achieve this goal is to decrease the length of the representation comparing to the original data. Decomposition should decrease the redundancy and correlation of symbols (pixel values) in the stream encoded, Before the quantization proceed. A syntax of decomposition with simple quantization results in very good efficient with much lower complexity and encoding/decoding time.

There are many different ways to perform the decomposition, the most popular are:

- Frequency transforms,
- Wavelet transforms,
- Fractal transforms.

The quantization reduces the number of symbols of the alphabet, which will be used by the intermediary representation of the encoded stream. So the information carried by the image is partially lost in this phase.

Adjustment of information loss level done when Compression methods often allow adjusting the level of information loss when the entropy is lower than encoded stream length. As decomposition determines the compression ratio,quality of the recovered image and size of information loss during encoding, it's the most important phase in all practical realizations of lossy compression.

there are two types of quantization in lossy compression methods: **Scalar Quantization and Vector Quantization**. the elementary unit of symbols for processing is the difference between both. In scalar quantization, this unit is equivalent of single symbol. While in vector quantization, it consists of some number of successive symbols a vector of symbols.

Both of these methods can employ regular or irregular length of intervals.

The adaptation manner of compression can go forward or backward. In forward adaptation, the input stream is divided into pieces, which have similar statistical characteristics, e.g. variance. It results better quantization of the entire input stream with cost of greater computing complexity and enlargement of the size of description of the quantization attached to the encoded stream.

In backward method of adaptive quantization builds the quantization based on data that processed during the quantization.In this method it would not needed

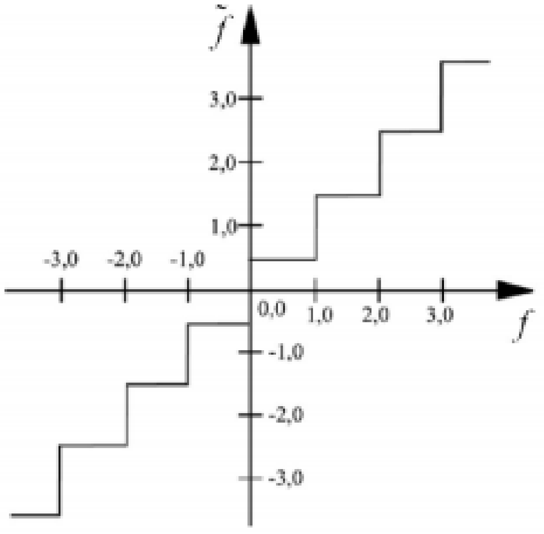


Figure 4.2: Regular scalar quantization [11]

to any additional information about the quantization to be attached to the encoded stream.

The last phase of lossy compression methods completes lossless compression method to which the output of quantization is passed as the input stream to be encoded. Most of the lossless methods are used in different lossy compression methods. Any type of lossless method can be used here, but it must be chosen with respect to the decomposition and quantization techniques.

Any phase of that we described in above scheme can be static or adaptive. Adaptive version usually increased effectiveness with the higher cost.
Compression ratio in lossy techniques is not limited by the entropy of the original stream. By higher compression ratio entropy of the encoded stream can be reduced.

Rate distortion theory which answers that what is the minimal entropy within the enough encoded stream to reconstruct the original image without exceeding a given distortion level. Notation, which will be used to explain the rate distortion theory, is explained on figure, In the figure bit rate is marked with R. This theory shows what the boundaries of compression ratio in lossy compression

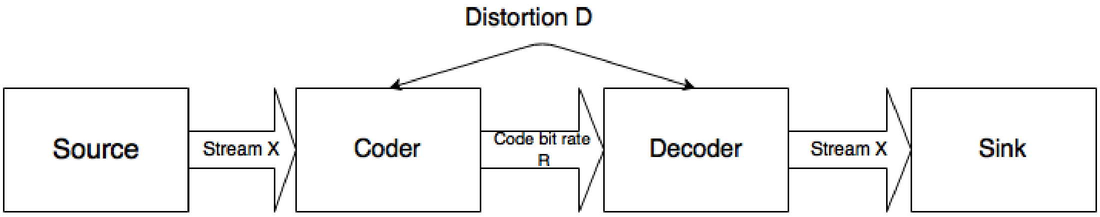


Figure 4.3: Compression system model in rate-distortion theory

methods. According to rate distortion theory the bit rate BR (average bit length per symbol) is related with distortion by following dependency:

$$BR(D_{max}) = \min_{d(X,\widetilde{X})} I(X, \widetilde{X}) \tag{4.1}$$

The I_m in above equation means "mutual information", it is the average information that random variables here $(X, \widetilde{X})$ convey about each other:

$$I_m(X, \widetilde{X}) = H(X) - H(X|\widetilde{X}) = H(\widetilde{X}) - H(\widetilde{X}|X) =$$

$$\sum_{x_i}^{\overline{X}} \sum_{\widetilde{x}_i}^{\overline{\widetilde{X}}} f_{X,\widetilde{X}}(x_i, \widetilde{x}_i).log\frac{f_{X,\widetilde{X}}(x_i, \widetilde{x}_i)}{f_X(x_i).f_{\widetilde{X}}(\widetilde{x}_i)} =$$

$$\sum_{x_i} \sum_{\widetilde{x}_i} f_X(x_i).f_{\widetilde{X}|X}(\widetilde{x}_i, x_i).log\frac{f_{\widetilde{X}|X}(\widetilde{x}_i, x_i)}{f_{\widetilde{X}}(\widetilde{x}_i)}$$

The random variable X describes the original data set and $\widetilde{X}$ represents the reconstructed dataset. the $f_X(x_i)$ represents the occurrence probability of determined symbol. The $f_{\widetilde{X}|X}(\widetilde{x}_i, x_i)$ is the conditional probability given symbol will occur in source $\widetilde{X}$ under condition that some symbol will occur in source X. Values $f_X(x_i)$ are defined by the statistics of the information source but the values $f_{\widetilde{X}|X}(\widetilde{x}_i, x_i)$ characterize the compression method. The mutual information has following properties:

$$0 \leq I(X; \widetilde{X}) - I_m(\widetilde{X}; X)$$

$$I_m(\widetilde{X};X) \leq H(X)$$

$$I_m(\widetilde{X};X) \leq H(\widetilde{X})$$

The distortion per symbol can be measured with Hamming distance or other measure:

$$d(x_i, \widetilde{x_i}) = (x_i - \widetilde{x_i})^2$$

or

$$d(x_i, \widetilde{x_i}) = |x_i - \widetilde{x_i}|$$

Independently from the measure that will be chosen the distortion d has fallowing properties:

$$d(x_i, \widetilde{x_i}) \geq 0$$

$$d(x_i, \widetilde{x_i}) = 0 when x_i = \widetilde{x_i}$$

The value D expresses the average distortion for an image and it is expressed with the equation:

$$D(X, \widetilde{x}) = Ed(X, \widetilde{x}) = \sum_{x_i} \sum_{\widetilde{x_i}} {}_{X,\widetilde{X}}(x_i, \widetilde{x_i}).d(x_i, \widetilde{x_i})$$

The formulas presented above state that, under the criterion that the average distortion will be not greater than the given value D_max, the minimal bit rate is equal to the greatest lower bound of the average mutual information. To find such compression method, characterized by the $f_{\widetilde{X}|X}(\widetilde{x_i}, x_i)$, one has to minimize the amount of information about random variable X carried by random variable $\widetilde{X}$ for given distortion level D not greater than D_{max}.

The relationship between bit rate and distortion level is visualized :

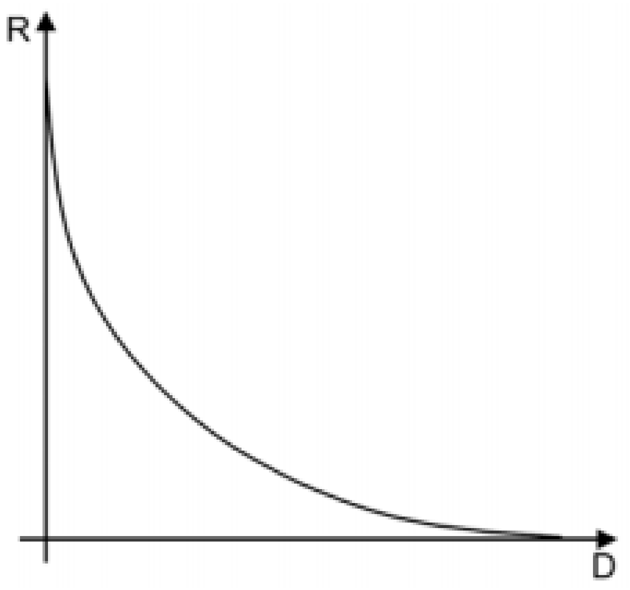


Figure 4.4: The relationship between bit rate and distortion in lossy compression [4].

4.2 Compression in Medical and Mammography Images

Medical images have also limited bit depth (how many bits are destined for description of single pixel color). X-ray images have bit depth equal 12 bit and ultrasound images only 8 bits. The matter is not so clear with Magnetic Resonance images. Image format used here can store 2^{16} (bit depth equal 16) tones of gray but, in fact, there are much fewer tones about 2^{9} (bit depth equal 9)[4].

There are also more important issues, which distinguish medical images from other. Medical images create a particular class of digital images, where the information carried by them is extremely important. High fidelity of compression and any other processing is required or the diagnosis could be erroneous. The loss of information may mislead not only when a physician personally examines the image but also when software is used for analyzing the image.

The receiver operating characteristic (ROC) analysis is an evaluation method used to measure the quality and diagnostic accuracy of medical images. It is performed by trained observers who rate the perceptible loss of information. The analysis gives for different medical image types the maximal compression ratios at which the fidelity of the images meets the expectations of the observers. For sample image types, the ratios are [12, 13]:

- Ultrasonography: 9 : 1
- Chest radiography: 40 : 1, 50 : 1 80 : 1 (JPEG2000)
- Computered Tomography: 9 : 1 (chest), 10 : 1 20 : 1(head)
- Angiography: 6 : 1
- Mammography: 25 : 1 (JPEG2000)
- Brain MRI: 20 : 1

The information loss should be avoided during processing but also very important is the quality of presentation of the image, especially the most important details. One should care about the faithfulness of image not only when it is presented in scale 1:1.

Due to small resolutions of medical images, their psychical size on a display device also will be rather small. Because of this, it is difficult to perform measurements by hand during diagnosing or even to read the image by a physician. Thus, magnification of the image is often very desirable and this means that also a zoomed-in image should be maximally true, legible and clear.

If it would be sure that images will not be magnified, probably the best choice for a compression method would be one of lossless methods. This group of compression techniques assures that no information will be lost during encoding and decoding processes; this means that the recovered image from a compressed file will be exactly the same as the original image.

The fractal compression has one large advantage over lossless methods it enables fractal magnification that gives much better effects that traditional magnification algorithms, e.g. nearest neighbor, bi-linear interpolation or even bicubic interpolation. Fractal magnification is actually the same process as fractal compression the image encoded with fractal method can be decompressed to arbitrary given size. An image compressed with one of lossless methods must be undergone to an interpolation algorithm if it has to be magnified. This means that although the compression algorithm did not cause any distortion to the image the interpolation algorithm will cause some faults. For example,block effect appearance, image pixelisation or image blurring. Fractal compression makes possible to keep the distortion rate on much lower level and the image remains sharp regardless of the size to which it is magnified.

Fractal magnification is not the only quality of fractal compression. As opposed to most of other compression methods, the fractal coding is asymmetric. From one hand, it is a drawback because encoding lasts much longer that in other methods. But at the same time it is an advantage because the decoding process is very fast it takes usually less time to decode an image with fractal method than to read the same image, but uncompressed, from the hard drive. This feature is useful when the image must be sent through the Internet the transmission time will be shorter because the image representation is shorter when is encoded with fractal method (lossy algorithm) than any lossless method, and there will be no significant additional time costs caused by decoding.

Another feature of fractal compression that attracts ones attention is the greatness of compression ratios that can be achieved with this method. Since it is a lossy method, it gives much smaller compressed file than any lossless compression algorithm. However, the medical images cannot be compressed with too high compression ratio because the loss of information can turn out to bee too high.

4.3 Fractal Compression Methods

Fractal compression methods, which belong to lossy methods, distinguish themselves from other techniques by a very innovative theory. To some extend, fractal compression diverges from the described above basic scheme of lossy compression methods.

(a) The picture of Lenna. (b) Selfsimilarity in Lenna's picture.

Figure 4.5: Self similarity in real images [14].

The most important part of this theory is that parts of an image are approximated by different parts of this image (the image is self-similar). This assumption makes possible to treat the image as a fractal.

According to B. Mandelbrot [14],the father of fractals, a fractal is A rough or fragmented geometric shape that can be subdivided in parts, each of which is (at least approximately) a reduced/size copy of the whole.

Fractal is a geometric figure with infinite resolution and some characteristic features. First of them is already mentioned self-similarity. Another one is fact that fractals are described with a simple recursive definition and, at the same time, it is not possible to describe them with traditional Euclidean geometry language they are too complex. As a consequence of the self-similarity of fractals, the fractals are scale independent change of size causes generation of new details. The fractals have plenty of other very interesting. Nevertheless, they are not necessary to understand fractal compression theory and they will not be explained here.

The essence of fractal compression is to find a recursive description of a fractal that is very similar to the image to be compressed. The distance between the image generated from this description and the original image shows how large information loss is. Although fractal compression is based on an assumption that the image can be treated a fractal, there are some divergence from above-presented fragments of fractal theory. In fractal compression self-similarity of the image is loosen it is assumed that parts of the image are similar to other parts and not to whole image.

All other properties of fractals remain valid for an image encoded with a fractal compression method. The image can be generated in any size, smaller or larger than the original. Quality of reconstructed image will be the same in all sizes, and edges always will have same sharpness. The number of details can be adjusted by changing the number of iterations for the recursive description of the image.

The fractal theory says that the recursive description of complex shape shall be simple. Any photographic-like image is very complex and if this image can be described as a fractal then a great compression ratio shall be achieved. The fractal description of an image consists of a system of affine transformations. This system is called fractal operator and has to be convergent.

In this section as there are lots of similarities between methods that briefly described in the last chapter, differences between the compression methods will be discussed. One has to keep in mind that many different fractal methods, elaborated by different authors, may implement some element in the same manner. The elements of the fractal compression algorithm that vary among different methods are grouped into several categories. Each section in this chapter corresponds to one such category.

4.3.1 Partitioning Method

Partitioning method will influence the parameters that evaluate the accuracy and quality of constructed image. For example, there is no need to attach any information about partitioning of fractional code in uniform partitioning. In the other hand, some methods need more than 44% of the fractal code to describe the partition [15], and some methods as Quadtree are between these two types. Quadtree partitioning takes about 3.5% of the total code size to define the partition [15]. Rate distortion performance cannot be affected by what we spoke from the three method that we mentioned negatively. largest space to specify the partition gives the best results. By this way, uniform partitioning plan impact on the number of transformations,it is the weakest one. the partitioning scheme has also impact on the number the partitioning scheme has also impact on the number of transformations. The Hartensteins method produces only few but large range regions,that two other method cannot achieved. Three partitioning methods, we mentioned above, are described in following subsections.

4.3.1.1 Uniform Partitioning

The most basic partitioning method option in fractional compression is uniform partitioning. uniform method is image independent as the ranges and domains have fixed size as 8×8 or 16×16. This partitioning method has some serious drawbacks. Firstly, it would be some details that size are smaller than the range. Moreover,as it is hard to find the domain with exactly same details, during the encoding, sort of details will be lost. The distance between 2 squares can be very small. f the ranges' size adjusted to minimize the first problem, this small ranges results more ranges that would effect the compression ratio efficiency in bad manner. Moreover, for some part of image we can have larger ranges by acceptable level of information that can result fewer transformation, So we will have better compression ratio.

4.3.1.2 Overlapped Range Blocks

Overlapped range blocks method is based on adjusting partitioning into squares that created Polidori and Dugelay [16]. As all ranges have same size $b \times b$ and domains $2b \times 2b$, it is very near to uniform partitioning. The difference is that the ranges are not disjunctive but mutually overlapping with half of their size. This means that all pixels belong to more than one range pixels close to the edge of the image belong to two ranges and the rest of the pixels are within four ranges. Partitions are encoded independently and decoding gives up to four different values for each pixel. From these four approximations, the final pixel value is calculated.

This method gives much better results than pure squares, e.g. effectively reduces the block effect and consume much more time. the image is four times encoded and four times decoded during each process. the fractal code representing the image is almost four times longer. The risk of losing small details is remained.

4.3.1.3 Hierarchical Approaches

Hierarchical approaches to image partitioning establish the first class of image-adaptive techniques. In decomposition of the image, compressed details are divided into smaller ranges and flat regions into large ones that depends on the content. This feature makes possible to overcome the limitations of fixed size (uniform) partitions scheme. There are two types of hierarchical approaches: **top-down and bottom-up.**

In **top-down approaches** in the beginning of encoding, whole image kept in single range or dividing into large uniform partitions. Depends on the method that used the range split, if it is not possible to find a domain that is close enough (error criterion) to a range then the range is being split into several ranges. In bottom-up approaches in order to have low level of information loss, it begins with dividing into small ranges. At this method, the neighbor ranges that are close enough to each other are being merged during later phase of partitioning, so the final ranges can have different size.

4.3.1.4 Quadtree Partitioning

The Quadtree partitioning presented by Yuval Fisher [17]was the first hierarchical approach to partitioning. All ranges have here the shape of a square. In this method, the set D of domains contains all square ranges with sides size 2, 4, 8, 12, 16, 24, 32, 48 and 64. in order to improve the quality of the encoded image, it can be admitted domains situated slantwise . In the top-down algorithm, the whole image is divided into fixed size ($3232 pixels$) ranges at the beginning.

In the next step, algorithm should find a domain (larger than the range) that gives the collage error smaller than some primary set threshold. If this attempt ends with failure for some ranges then each such range is divided into four.This procedure repeated for all newly created ranges, i.e. fitting domains are being searched for ranges and, if necessary, the non-covered ranges are being broken down. The encoding continues till there are no ranges that remain uncovered or the size of the ranges reaches a given threshold.

In the second case, the smallest ranges are paired with domains that do not meet the collage error requirement but are closest to corresponding ranges. If unions of quadrants created during division of a range introduce, it can increase adaptivity of the division that can improve results.

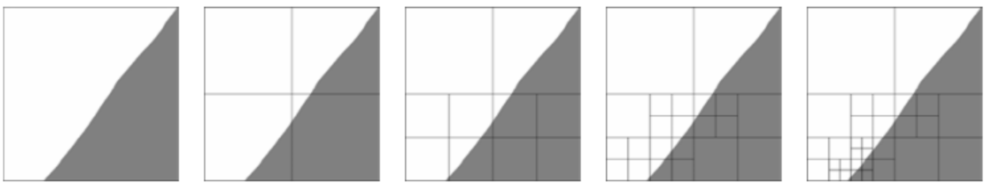


Figure 4.6: Quadtree partitioning of a range. Four iterations [17].

As disadvantage of Quadtree partitioning method we can note that, all ranges are divided in the same way, independently from the content of the ranges. The size of ranges and the structure of partitioning are adaptive to the whole image but as four quadrants of the input range the act of braking down a single range produces always the same output.

If partitioning process was adaptive also at the stage of drawing the borders of future regions during range block division, it would be better fitted to the content of the image. This improvement would result in larger range blocks, i.e. in less number of transformations.

4.3.1.5 Horizontal-Vertical Partitioning

In the horizontal-vertical (HV) partitioning method [17], the shape of a range can be not only a square but also any other rectangular because a range (when there is no domain close enough to it) is divided into two rectangles instead into four squares. The frontier between the two rectangles is established in the most significant horizontal or vertical edge. Thus, this method is an answer to disadvantages of Quadtree partitioning it tries to find best division of a range into two new ranges by horizontal or vertical cut.

The image is partitioned in this manner from the beginning, i.e. there is no initial phase in which the image is divided into uniform partitions (like in Quadtree partitioning). The algorithm includes also mechanisms preventing from degeneration of rectangles.

The algorithm uses two formulas ($v_n andh_m$) that allow determining the direction and the position of the cut:

$$v_m = \frac{min(m, width(R_i) - 1 - m)}{width(R_i)} .(\sum_{n=0}^{height(R_i)-1} r_{m,n} - \sum_{n=0}^{height(R_i)-1} r_{m+1,n}) \quad (4.2)$$

$$h_n = \frac{min(n, height(R_i) - 1 - n)}{height(R_i)} .(\sum_{m=0}^{width(R_i)-1} r_{m,n} - \sum_{m=0}^{width(R_i)-1} r_{m,n+1}) \quad (4.3)$$

where $width(R_i) \times height(R_i)$ is the dimension of range block R_i and $1 \leq m < width(R_i), 1 \leq n < height(R_i)$. The second factor pf these formulas, $(\sum_n r_{m,n} - \sum_n r_{m+1,n})$ and $(\sum_m r_{m,n} - \sum_m r_{m,n+1})$, give the difference of pixel intensity between adjacent columns (v_m, v_{m+1}) and rows (h_n, h_{n+1}). Maximal values of these differences point out the most distinctive horizontal and vertical lines.

the first factor $\frac{min(m,width(R_i)-1-m)}{width(R_i)}$ and $\frac{min(n,height(R_i)-1-n)}{height(R_i)}$ ensure that the rectangles created by splitting the range block will not be too narrow the closer a possible cutting line location is to the middle of the range block, the more privileged it is.

At this point, we have two alternative lines along which the split can be done one vertical and one horizontal. The HV partitioning allows cutting along only one of them:

- if $max(h_0, h_1, ..., h_{height(R_i)-1}) \geqslant max(v_0, v_1, ..., v_{width(R_i)-1})$ then the range block is partitioned horizontally.
- otherwise, the range block is partitioned vertically

In other words, the more distinctive cutting line is chosen from the two alternatives. The increased adaptivity is paid dearly with increased time complexity (due to the variety of range shapes and additional computations) and longer description of the partitions. However, these additional costs pay off the rate distortion is significantly improved comparing to Quadtree partitioning method. This superiority is caused by better adaptivity and larger range block sizes (i.e. lower number of range blocks).

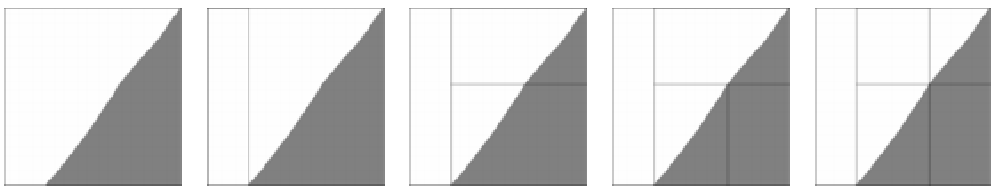


Figure 4.7: Horizontal-vertical partitioning of a range. Four iterations[17].

4.3.1.6 Triangular Partitioning

Next partitioning method [18] is based on triangles. In the first step, the rectangular image is divided into two triangles along one of diagonals. At this point, the recursive algorithm begins. Each triangle, for which no suitable domain can be found, is divided into four triangular ranges. The borders between these triangles are drawn between three points that lie on three diverse sides of the range to be divided. The points that define the borders can be freely chosen in order to optimize the division and minimize the depth of the tree representing the partitioning, i.e. the number of transformations.

There was also elaborated a second triangular partitioning scheme, in which the triangular range is divided along a line from a vertex of this triangle to a point on the opposite side [19].

The triangular partitioning has several advantages over HV partitioning. First of them is the fact that distortions caused by not ideal matching of the ranges and domain are less noticeable. The second very significant advantage is possibility of occurrence of rotation angles within the transformations other than multiple of right-angle. This is because the triangular ranges can have any orientation and rectangular ranges (HV, Quadtree, fixed size partitioning) can lie only horizontally or vertically. The largest advantage of triangular partitioning is reduction of the block effect, which can be observed in uniform partitioning.

Nevertheless, this partitioning scheme has also some heavy drawbacks. The comparison of a domain block with a range block is hampered because of the difficulties with interpolation of the domain block when the pixels from these two blocks cannot be mapped one-to-one. This problem occurs in all partitioning schemes that are not based on right-angled blocks and is the reason why the right-angled methods are superior [20].

4.3.1.7 Polygonal Partitioning

The polygonal partitioning is very similar to horizontal-vertical but is more adaptive to the image. It was invented by Xiaolin Wu and Chengfu Yao [21] but Reusens was the one who applied it to fractal image compression [22]. In this method, a range can be divided horizontally, vertically (like in HV) or along a line inclined by 45 or 135 degrees.

Other method to get polygonal blocks is the modified Delaunay triangulation method in the merging phase of this method, not only triangles can be created but also quadrilaterals [23]. However, this method belongs to the second group of partitioning schemes the split-and-merge approaches.

4.3.2 Split-and-Merge Approaches

The hierarchical approaches perform the partitioning while the pairs of ranges and domains are being found. The split-and-merge approaches divide the image into partitions before the searching for transformations is started. The partitioning process consists here of two phases. The first phase the splitting yields a fine uniform partitioning or a partitioning with various density of ranges for different parts of the image. The second phase the merging combines neighboring ranges with similar mean gray levels.

4.3.3 Delaunay Triangulation

Delaunay triangulation was adapted to fractal coding by Davoine and Chassery [24, 25]. In this method, the partitioning results in a set of non-overlapping triangles that cover whole image. The splitting phase starts by dividing the image into regular, fixed size triangles. This triangulation is represented by regularly distributed points, which are equal to triangles vertex. Then the triangles are investigated and if any triangle is not homogeneous in sense of variance or gradient criteria then a point is added in the barycenter of the triangle. The splitting is recursively repeated until all triangles are homogeneous or the non-homogeneous triangles are smaller than a given threshold. Before each iteration, the triangles must be recalculated based on the set of points.

The merging removes certain vertex and by this action, the triangles are combined. A vertex is removed if all triangles to which it belongs have similar mean gray levels. Each single change of the set of vertex entails the necessity of recomputing the triangulation before following actions are performed.

The Delaunay triangulation has the same main advantages as the triangular hierarchical partitioning related with unconstrained orientation of triangles. However, the number of transformations determined with Delaunay triangulation is lower than in hierarchical approaches.

The triangles can be merged not only to larger triangles but also to quadrilaterals [23]. This increases the compression ratio because the number of transformations is smaller in such case. When the basic Delaunay partitioning and the enhanced scheme result in similar compression ratio then the quality of the reconstructed image is better in the quadrilateral approach.

4.3.4 Irregular Regions

The methods that produce irregular shaped range regions realize the splitting simply by utilizing the existing simple partitioning methods. The uniform partitions were employed in first algorithm based on irregular regions created by Thomas and Deravi [26] but also in the work of other researchers [27, 28, 29]. The Quadtree partitioning was introduced to irregular partitioning by Chang [30, 31]; Ochotta and Saupe also used this schema [32].

The small squares from first phase are merged to form larger squares or irregular range blocks. This partitioning scheme adapts very well to the content of the image, which is being encoded.

However, there are problems with concise description of the regions boundaries. There are two main approaches to this issue: chain codes and region edge maps. The chain coding describes the path that agrees with the boundaries. To specify this path a starting point and a sequence of symbols representing steps (simple actions: go straight, turn left, and turn right) must be stored in the fractal code. The length of the step is equal to the length of the side of region block

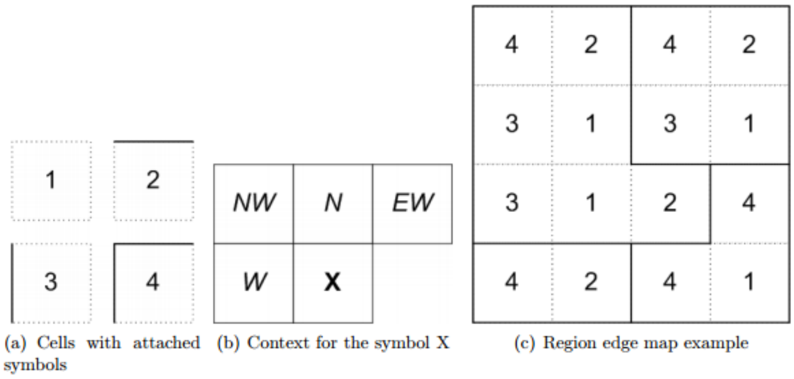


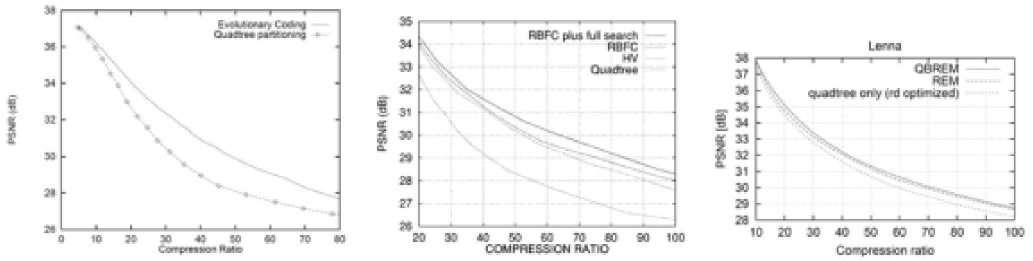
Figure 4.8: Region edge maps and context modeling [11]

in uniform partitioning, and it is equal to the length of the side of the smallest region block in the Quadtree. The most basic version of chain coding encodes each closed region boundary into one path with specified starting position. The performance of such approach leaves much to be desired because redundant information is present since almost all of the boundaries are sheared by two regions.

The region edge map [33] utilizes a grid of squares. If uniform partitioning was used in splitting phase then the grid is equal to these partitions. If Quadtree partitioning was used then the cells in the grid have the same size as the smallest ranges any range (of Quadtree partitioning) can be either a union of cells or a single cell. Each single cell is provided with one of four symbols that indicate whether (and where) there is a range boundary at the edge of the cell; the symbol is stored in two bits. There are only four instances considered:

- no range boundary
- boundary on the North edge
- boundary on the West edge
- boundary on the North and on the West edges

The region edge maps can be efficiently encoded with an adaptive arithmetic coding and context modeling. The context is build of four cells processed (encoded or decoded) before the current cell these are the neighbors in the West, North West, North and North East directions. There can be 256 different combinations of symbols in the context; some of these combinations indicate which symbols cannot occur in the currently processed cell. For example, when the symbol 1 or 2 is attached to the cell to the North and the symbol in the cell to the West is 1 or 3, then the current symbol cannot be 2 or 3 either. This fact allows shortening the fractal code.



(a) Chain code (b) Region edge maps (c) Quadtree-based region edge maps

Figure 4.9: Performance of compression methods with irregular partitions[27, 15, 32]

The irregular partitions guarantee good results of the encoding. Such partitioning schemes are ultra adaptive to the image content and since they are right-angled they are devoid of the drawbacks of triangular partitioning. The experiments (see figure 3.4) show that they outperform any other partitioning method. However, there is still disagreement which method is superior, what will be explained in the last section of this chapter.

4.3.5 Domain Pools and Virtual Codebook

The two terms the domain pool and the codebook are very close connected with each other. In the literature, they are often used interchangeably, but here by a domain pool, in the context of fractal coding, the author means a set of domains (a subset of all possible domains in the image) that is being used during searching for a matching domain for a range. The codebook blocks correspond to domain blocks but their size is the same as the size of the range. The set of all codebook blocks is called virtual codebook. The codebook in fractal compression is virtual because it is not needed at the decoding (it is not stored in the fractal code) it is used only during the encoding phase. Summarizing, the codebook denotes a set of codebook blocks, which are contracted (downfiltered) domain blocks from the domain search pool.

The length and contents of the domain pool (codebook) is crucial for the efficiency of the encoding process. If the domain pool is larger then more bits are required for the representation of selected domain in the code. At the same time, larger domain pool entails longer time for searching a domain for a range; this results in much extended encoding time. However, larger domain pool also has a positive effect it helps to achieve higher fidelity.

There are two main approaches to domain search that can be observed in different encoding methods. The first one, called global codebook search, provides the same domain pool (codebook) for all ranges of the image but there may be various the domain pools and codebooks for different classes of ranges. Local domain pool search, the second approach, makes the codebook dependent on the position of the range.

4.3.5.1 Global Codebooks

This solution to domain search is based on an assumption that a range and a domain can be paired into a transformation even if they lie in completely different parts of the image. This assumption is confirmed by [34, 35, 36] where authors state that there cannot be determined a neighboring area of a range within which the best domain for the range lies.

An example of a global codebook can be seen before. In the example fractal encoding algorithm, each domain block of the image is considered during the searching of matching domains and ranges. Because the algorithm employs uniform partitions, the domain pool consists of blocks of same size. The interval between corresponding borders of neighboring domain blocks is equal to one pixel vertically or horizontally. This solution is very complex computationally due to large number of blocks within the codebook. The time cost is here very high but this procedure gives optimal loss of information because the best matching between a range and a domain will be always found.

In order to reduce the time cost larger intervals between blocks, which are appended to the domain pool, are introduced. The literature gives two typical interval values: equal to the domain-block width or to half of the domain-block width. This simple move significantly decreases the number of domains in the pool and, thanks to that, speeds up the searching for a domain. The main rule is that the larger the domain pool is the better fidelity is achieved but with higher time cost. So reducing the size of domain pools gives shorter searching time (and shorter encoding time), but more information is lost (the errors between paired ranges and domains might be larger). Higher intervals between the domains in the pool result also in better convergence at the decoder (less iterations are required to decode the image).

The global codebook constructed like above can be used when the image is segmented into uniform range blocks or with Quadtree partitioning. It can be also used with HV partitioning, but a domain pool containing ranges (larger than currently processed range) or blocks created by the partitioning mechanism (used also for determining range-blocks) are more often used. These two last methods of constructing global domain pools can be also used with other adaptive partitioning schemes. In the Quadtree scheme there is not one global domain pool but several for each class of ranges (all ranges within one class have same size) is provided a separate domain pool (and codebook) that contains domains twice as large as ranges within the class.

4.3.6 Local Codebooks

A number of researches [37, 38, 39] have proven that the probability density of the spatial distances between ranges and matching domains has a distinct peak at zero. This means that it is much more likely to pair a range with a domain that is close to the range than with a distant one.

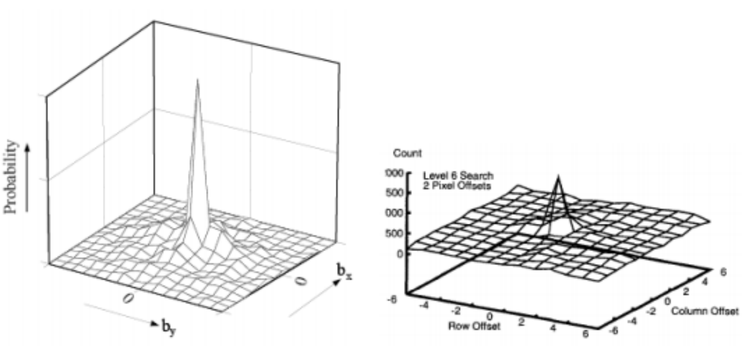


Figure 4.10: Probability density function of block offset.[39, 38]

4.3.6.1 Restricted Search Area

In fact, the probability that a distant domain will be judged as a matching one is so small that the searching can be restricted to only spatially close domain blocks. The remaining part of search algorithm remains unaffected.[37]

4.3.6.2 Spiral Search

In the approach the search order is modified - the codebook blocks that are more likely to provide a good match for currently processed range block are tested first. Therefor, the search is performed along to spiral-shaped path, which has a beginning in the codebook block directly above the range block and gradually recedes from the range. The search area can be here restricted by defining maximal number of range blocks that shall be tested for each range block the length of the path.[40]

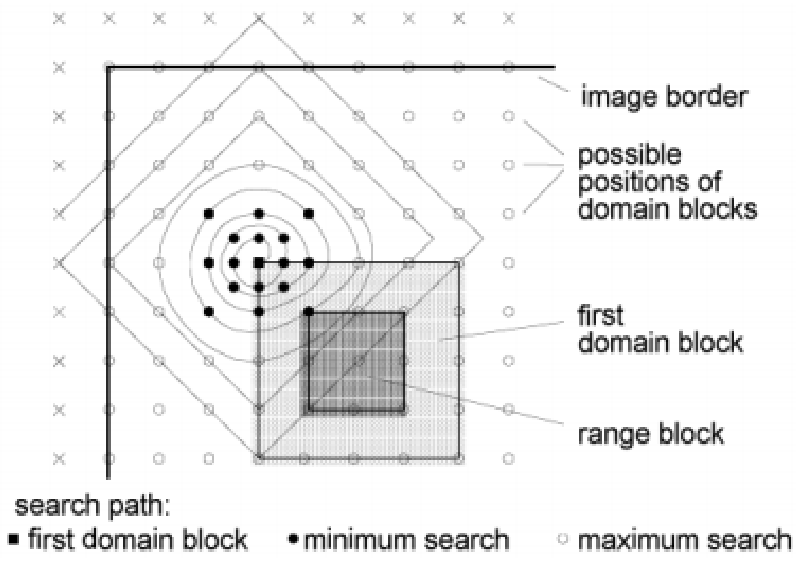


Figure 4.11: Spiral search [40]

The literature gives several ways in which the advantage of this fact can be taken.

It can be noticed that the density of the domain blocks tested during the spiral search is higher at the beginning of the path (close to the range block).

4.3.6.3 Mask

Another way to determine a not numerous domain pool is to put a mask on the image and center it at the currently processed range block. The mask indicates the locations of domain blocks that should be included into the domain pool. These locations are denser near to the center of the mask and condensation decreases with the increase of the distance to the center.[39]

4.3.6.4 Solutions without Domain Search

There are several ways to eliminate the time-consuming domain search. The first of them pairs a range and a domain when the position of the domain block fulfills some conditions. For example, P. Wakefield [41] proposes to pair domains with ranges in such manner, that the range block lies within the domain block and the dominant edge should be in the same relative position in both blocks. Other solutions force the matching domain to be in a fixed relative position to the range [42] or restrict the domain pool to a very small set of domains neighboring to the range [43].

Because this class of fractal methods eliminates one of the most time-consuming phases, the encoding is very significantly accelerated. At the same time, the search-free methods give best rate-distortion results [38] . However, in medical imaging the information carried by the image is much more important than the achieved compression ratio and the search-free methods loose details by imprecise matching of domains and ranges. But without any doubt, it can be said that local codebooks outperform global ones what have been proved in [39] where the signal-to-noise-ratio were only 0.3 dB lower for the search with a mask than for a full search; at the same time the domain pool contained only 12% of the domains from the global pool.

4.3.7 Classes of Transformations

As it was already said, the transformations determined during encoding have to be affine and contractive. However, this restriction is very weak and further limitations have to be introduced in order to provide full automation of the encoding process. Thus searching for the transformations that will constitute the fractal code is performed only within a limited class of affine transformations. The choice of this class influences the effectiveness, fidelity of the algorithm and convergence properties of the fractal operator. Thus, the importance of selecting the right class of transformations cannot be overrated since it is crucial for both process of compression encoding end decoding.

A transformation usually can be decomposed into three separate transformations that are carried out one after another. Therefore, a single elemental block transformation τ_i (from the domain block D_i to the range block R_i) is a composition of three transformations:

$$\tau_i = \tau_i^I \circ \tau_e^S \circ \tau^C \tag{4.4}$$

After the transformation τ_i is placed on the domain block D_i , the resulting pixels may be copied into the range block R_i . Thus, transformation τ_i

is the key part of the affine transformation w_i , which maps domain block D_i on to range block R_i. In order to transform a domain block into an appropriate range block firstly the domain block is spatially contracted (transformation τ^C), the product of this phase is a codebook block. The order of pixels within one codebook block is deterministically changed by τ_e^S i.e. it is undergone one of symmetry operations like rotation or reflection. The used symmetry operation is taken from a fixed pool;e denotes here the index of the used operation. The last component transformation τ_i^I is an intensity transform, which adjusts the brightness of the codebook block.

The contraction transformation usually is the same for all domains. However, the symmetry operation is known not before the searching for matching pairs of domains and ranges. Same situation is with intensity transformation, when a domain and a range are compared (during the searching), this transform is defined in such way that the error between them is minimized.

All domains D_k ($0 \leq k < \overline{D}$, where $\overline{D}$ length of the domain pool) from the pool are transformed by τ^C what gives the codebook of blocks C_k . The codebook can be expanded thanks to symmetry operations every block of the codebook is transformed by all symmetry operations and the products of these operations are included in the codebook. Theoretically, this step should allow better matching between the codebook block and the range block (during searching a domain block fitting to a range block). One has to keep in mind that the codebook is virtual, i. e. the codebook blocks are not stored in four copies that differ from each other only with the rotation angle there is a single copy of a codebook block that is rotated during the search.

Then the real search is being performed. For a range block R_i every codebook block C_k s checked the coefficients of the intensity transformation (that minimize the distance between the codebook block and the range block) are calculated, i.e. the transformation τ_{ik} is being determined. From all of the τ_{ik} (and, at the same time, from all of the codebook blocks) the one is picked that gives the minimal error between the range block and the product of the transformation the chosen transformation becomes τ_i.

The description of the contraction transformation τ^C an be sawed into the program the transformation is the same for all domains/ranges and the same for the encoder and the decoder. But information about the τ_e^S and τ_i^I has to be attached to the fractal code. In particular, the symmetry operation musts be pointed out and the coefficients of the intensity transformations must be stored for every range block.

4.3.8 Spatial Contraction

The spatial contraction of domain is not necessary for the process of fractal compression. The transformation must be counteractive but the metrics that are used to assure the contraction usually is not influenced by the spatial dimension [25, 34]. A sufficient constraint is that a domain block and a range block paired into one transformation cannot be equal. However, the spatial contraction is commonly used in almost fractal compression methods. It was introduced by Jacquin [44] and it is moved out directly from the first fractal compression algorithm where the spatial size of square domain blocks was twice as large as the size of range blocks.

Also using the same contraction ratio as Jacquin became a custom the spatial contraction usually reduces the dimensions of a domain block by two. However, it is possible to adjust this number in order to achieve desired behavior of the encoder or decoder. A contraction ratio higher than 2 : 1 decreases the number of iterations needed to reconstruct the image from fractal code (fractal operator)

[45]. It is possible to adjust the contractivity in such way that the decoding will be made by a single iteration [35]. A contraction ratio smaller than 2 : 1 entails higher error propagation during decoding. But it also has positive effects it allows better approximations of range blocks with codebook blocks [40]. In the original work of Jacquin [44], the domain block was contracted by averaging of four neighboring pixel values into one. So according to this, when the width and the height of the codebook block are equal to h and the contraction is made by factor of 2 then a value of a pixel of a codebook block C_i can be calculated from the following formula:

$$C_i(m,n) = \frac{D_i(2m,2n) + D_i(2m+1,2n) + D_i(2m,2n+1) + D_i(2m+1,2n+1)}{4} \tag{4.5}$$

for all $m, n \in 0, ..., h-1$. This formula can be easily generalized to any size of the codebook block.

The contraction by neighboring pixel averaging is still very popular but also other solutions can be employed here. Barthel and Voye introduced anti-aliasing filter what allowed to obtain better coding results [40]. Instead of averaging neighboring pixels, the excess pixels can be removed. This solution slightly speeds up encoding but has negative influence on the accuracy [17, 35].

4.3.9 Symmetry Operations

The symmetry operations, called also isometries, operate on pixel values of a block without changing their values. They change the positions of pixels within the block in a deterministic way. For a square block, there are eight canonical isometries[44]:

1. identity
2. orthogonal reflection about mid-vertical axis of block,
3. orthogonal reflection about mid-horizontal axis of block,

4. orthogonal reflection about first diagonal ($m = n$) of block,
5. orthogonal reflection about second diagonal of block,
6. rotation around center of block, through $+90°$,
7. rotation around center of block, through $+1800°$,
8. rotation around center of block, through $-90°$.

The isometries significantly enlarge the size of the domain pool so they should take effect in better fidelity of the reconstructed image. According to a number of researchers, all isometries are used in same frequency during encoding[36, 46]. This proves that they are useful and fulfill their destination. At the same time, other authors prove that the isometries are dispensable and have no positive effect [37, 47, 46]. Probably different design choices not directly related with the isometries are the main cause of this contradiction [20]. However, an overwhelming agreement can be observed in the literature, that the use of the isometries results in weaker rate-distortion relation [46, 48, 49, 50]. Besides, other affine transformations can be used in place of isometries [47].

4.3.10 Block Intensity

The last component transformation also operates on pixel values but it changes the luminance of pixels instead their positions. Once again, the most basic intensity transformation was introduced already by Jacquin. It is linear and operates on one codebook block (after application of symmetry operations) and one block of unit components:

$$C_i{}' = s_i C_i + o_i 1 \tag{4.6}$$

The s_i and o_i denote the scaling and offset respectively. These coefficients are calculated by the encoder when the best approximation $R \approx s_k c_k + o_k 1$ is found ($0 \leq k < \overline{C}$, where $\overline{C}$ length of the codebook).

Although the linear intensity transformation still can be found in many more present fractal compression methods, other transformations can be found in the literature. According to the authors, these new approaches improve the fidelity of the compression by enabling better approximation of a range block by a codebook block.

4.3.11 Orthogonalization

Oien [51] modified the intensity transform by introducing orthogonal projection prior to scaling. From the codebook block the dc component is being subtracted. The dc denotes the mean pixel value of the codebook block:

$$d_c = \frac{C_i^1 + ... + C_i^{\overline{C_i}}}{\overline{C_i}} \tag{4.7}$$

where $\overline{C_i}$ is the number of pixels in a codebook block.

The intensity transform in this case can be described by following formula:

$$C_i{}' = s_i(C_i - \frac{\langle C_i, 1 \rangle}{\|1\|^2}) + o_i 1 \tag{4.8}$$

The $\langle C_i, 1 \rangle$ is the inner product of the codebook block and the block of fixed intensity $\|1\|$ is the derived norm of an appropriate product space, i.e. L_2 here.

this transformation yields a block that is orthogonal to the block of unit coefficients 1 and gives several advantages. First of all the s_i and the o_i coefficients are decorrelated. When a special choice of domain pool is made (each domain is a union of range blocks in Quadtree partitioning, the contraction based on pixel averaging), the decoding is accelerated the convergence of the decoder is guaranteed to be in a finite number of iterations. The number of iterations is independent of the s_i and o_i coefficients and only the sizes of the domains and ranges influence it [52].

4.3.11.1 Multiple Fixed Blocks

The topic of multiple fixed blocks was raised by Oien, Lepsoy and Ramstad [53] and continued by Monro[**?**, 54] and many other researchers. The main idea is based on replacing the single fixed block 1 with multiple fixed blocks V_h:

$$C_i{}' = s_i C_i + \sum_h o_{ih} V_h \tag{4.9}$$

4.3.11.2 Multiple Codebook Blocks

Another approach uses several codebook blocks that are independently scaled:

$$C_i{}' = \sum_h s_{ih} C_{ih} + O_i 1 \tag{4.10}$$

It is also possible to merge the multiple fixed blocks approach with the multiple codebook blocks approach. In this case, also domains that do not have to be spatially contracted can be used.[55, 56, 57, 58, 59] The linear combination multiple domain blocks and multiple fixed blocks was used in [57]and resulted in great rate-distortion relation at the bit rate 0.43, the peak signal to noise ratio achieved 34.5 dB .

4.3.11.3 Polynomials

Other attempt to the intensity transformation [Mon93a, Mon94b, Mon94a] resigns from the linear character and uses higher order polynomials. When the transformation is a second order polynomial then an additional component is added to Jacquins basic transformation the codebook block with quadratic form:

$$C_i{}' = [c' \mid s_{i2} c^2 + s_{i1} c + o_i 1] \tag{4.11}$$

where c symbolizes a matrix coefficient of Ci and c' a coefficient of C_i'. The third order polynomials will require extending the transformation with one more component codebook block with cubic form:

$$C_i' = [c' \mid s_{i2}c^3 + s_{i2}c^2 + s_{i1}c + o_i 1] \quad (4.12)$$

When the basic linear transformation is used then a single pixel of the codebook block is undergoes the following intensity transformation:

$$\tau_i^I = s_i z + o_i \quad (4.13)$$

The application of the polynomials modifies the shape of the fractal operator [60] . Here the operator takes following form:

$$w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & \tau_i^I \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ 0 \end{bmatrix} \quad (4.14)$$

The τ_i^I (intensity transformation) looks as follows:

- order 2 polynomials

$$\tau_i^I(z) = s_{i2}z^2 + s_{i1}z + o_i 1 \quad (4.15)$$

- order 3 polynomials

$$\tau_i^I(z) = s_{i3}z^3 + s_{i2}z^2 + s_{i1}z + o_i 1 \quad (4.16)$$

Of course, also higher order polynomials can be applied but it results in worse compression ratio because more parameters have to be encoded. However, the higher order polynomials are used the better fidelity can be achieved [61]. The use of second order polynomials turns out to be the best when it comes to the rate-distortion relation [38].

4.3.12 Quantization

The quantization occurs in encoding as well as decoding. During the encoding, the scaling and offsets coefficients have to be quantized. The domain positions, the description of used symmetry operations and any partition description relevant to the adaptivity of the segmentation are represented by discrete values from the beginning.

4.3.13 Quantization During Encoding

Most often, a uniform quantization is used. Nevertheless, the distribution of the scaling or of the offset coefficient in general has a strongly non-uniform character. The application of a uniform quantization method entails inefficiency and entropy compression of quantized coefficients can be very useful for eliminating it.

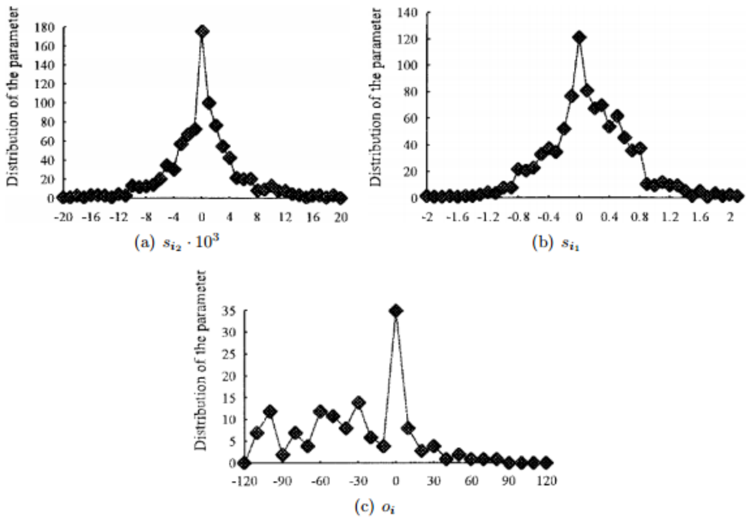


Figure 4.12: Distributions of scaling and offset coefficients (second order polynomial intensity transformation)[62]

The quantization occurs also during decoding. Each iteration of the algorithm produces an image that is an approximation of the fixed point of the IFS. In the original approach, the images created in successive iterations were stored as raster images, i.e. the pixel values were quantized. However the brightness of the fixed points pixels takes real values and not discrete and the error caused by quantization in this solution is propagating on the result of following iterations. This may cause difficulties with reaching the correct values of brightness of some pixels. This problem can be minimized by introducing matrices of real numbers to represent the images created in successive iterations. This solution is called the Accurate Decoding with Single-time Quantization and guarantees that the quantization will be performed only once when the matrix from the last iteration will be converted to a raster image [60].

4.3.14 Decoding Approaches

The fractal code contains the quantized coefficients of the fractal operator. The decompression is actually the process of computing the fixed point described by this operator. The fractal operator is independent of the size of the original image so the decoding may result in a reconstructed image in any size the image may be zoomed in or zoomed out comparing to the original one. The basic decoding algorithm is based on PIFS and was already explained. One of advantages of fractal compression is fast decoding usually it takes less than 10 iterations. However there are introduced some alternative approaches that improve the speed or accuracy of the process.

4.3.14.1 Pixel Chaining

The method can be utilized only when the intensity transform is based on sub sampling. In such situation, each pixel of a range block is associated with one reference pixel in the domain block the range and the domain are paired by a transformation. The reference pixel lies not only in the area of the domain block but also in the area of some other range block. Thus, another reference pixel is associated with it. In this way, a chain of pixels that are associated is created. The pixel chain can be used in two manners. The first way is utilized it to track the path of influence in order to find a pixel with wanted value. The second way executes a part of the chain long enough to achieve acceptable pixel value[35, 47].

4.3.14.2 Successive Correction Decoding

The basic decoding algorithm uses for each iteration a temporary image in which the changes are made by transformations. This means that the image that provides the virtual codebook in current iteration remains unchanged by the transformations and the range blocks are situated on the temporary image.

The successive correction method is inspired by Gauss-Seidel correction scheme. The basis of the successive correction algorithm is resignation from the temporary images the transformations operate on the same image. The domain blocks covering actually decoded range blocks are immediately updated, i.e. the change made by one transformation is visible for transformations executed after that one but in the same iteration.

The main advantage of this technique is increased decoding speed. A further speed improvement can be made by ordering the transformation. In the image are staked out domains with different density. Transformations that have domain ranges in areas of the highest domain concentration are executed first in each iteration [63, 64].

4.3.14.3 Hierarchical Decoding

The first stage of hierarchical decoding is actually nothing else as the baseline decoding algorithm. The only difference is that the image is reconstructed at a very low resolution the size of the range blocks is reduced to only one pixel. This low-resolution image is treated as a basis to find the fixed point in any other resolution with a deterministic algorithm (similar like in wavelet coding the transformations from domains to ranges are treated as consecutive resolution approximations in the Haar wavelet basis). Because vectors of lower dimensions are processed during IFS reconstruction, there are considerable computational savings comparing to the standard decoding scheme[65, 66, 67].

4.3.14.4 Decoding with Orthogonalization

his approach was already mentioned. It requires some changes of the encoding process, i.e. all domain blocks from the pool have to consist of a union of range blocks and the intensity transform has to utilize orthogonalization. These restrictions result in meaningful benefits: an uncomplex computationally decoding algorithm based on pyramid-structure, decoding length independent of the transformation coefficients (it depends only on domain and range sizes) and at least as fast as in the basic scheme [52].

4.4 Post-processing

Any fractal compression method is based on blocks and, because of this, block artifacts are always present in the reconstructed image. In order to reduce the undesired artifacts the reconstructed image can be post-processed. The block boundaries are subjected to smoothing [35, 47].

There are at least several ways to reduce the block artifacts during post-processing. The first one is simply the right choice of partitioning method the overlapped ranges give very good result, the blocks are also less noticeable when a highly adaptive partitioning method is used.

A simple method that uses a lowpass filter can be engaged. However, the results are not satisfying[35] . Other estimation-based methods, more complex, give better performance [68, 4] .

There are also post-processing methods that depend on the partition scheme used in the compression and heads for the best overall performance taking into consideration the human visual system. The Laplacian pyramidal filtering presented in[61] is an example of such method.

4.5 Discussion on Methods

As we saw in chapter, at the same time the diversity of issues connected with building a fractal compression methods. Although the basis of fractal compression remains the same in all implementations, there still is notable latitude during the act of constructing a fractal compression method because there are no standards to it and only a general idea how to utilize the fractal theory to image compression. This freedom can be problematic because there is not always agreement which solutions in particular elements of the fractal compression method yield the best effects. This confusion is being amplified by the fact that each design decision influences on the performance of other design elements.

As an example, the choice of partitioning scheme can be given there there is a disagreement in the literature which one is the best. Some researchers appoint the simple Quadtree scheme as the superior comparing to polygonal and HV partitions [22] . Others, at the same time, prove that the HV partitioning gives better results than the Quadtree [36, 15, 29]. However, most of the researchers agree that irregular regions give better results in rate-distortion sense over Quadtree scheme [32, 27, 30, 29, 15, 69] . The comparison of HV with irregular schemes does not show as large superiority of the method based on irregular regions [15] , especially the methods utilizing Quadtree partitioning in the split phase [30, 31, 32], or even these two approaches yield very similar rate-distortion performance[29]. One can notice that for small compression ratios, for which the best fidelity can be obtained, the HV partitioning results in slightly better Peak-Signal-to-Noise ratio. However, the irregular-shaped approaches allow encoding an image with the same PSNR ratio faster. A remarkable observation is that none of the partitioning schemes that are not based on right-angled regions matches to the performance of above-mentioned methods [20].

The effectiveness of fractal compression can be improved by merging it with transform coding or wavelets. Nevertheless, such hybrid-methods are not discussed in the document.

Chapter 5

Clustering

As we used K-Means Clustering in our Method we explain briefly about it in this chapter.

Sorting images in segments and meaningful points has important effect in images analysing as they contain many objects and features. Clustering is computational tool to do segmentation that tries to find specific patterns or structures in a data set. Objects in each cluster have a certain degree of membership. In another word, clustering is a process of dividing object (pixel) into groups based on its features. So, cluster means a set of similar objects (pixels), which they are totally different from objects belonging to other clusters.

Clustering algorithms are classified in two groups: **hard clustering algorithm**, **soft clustering algorithm**. In **hard clustering methods** data partitions to specific clusters and each data point belongs to just one cluster. **soft clustering algorithm** is responsible to associate membership levels and consequently locate each data points in one or more clusters [70].

Summarize of the clustering process is shown in sample below. The given data segmented into three clusters identifies by three different colors Red, Blue and Green.

Features that are based of classification of an images are like keyword and content of image. **Keyword based clustering** is to assign a font which describes

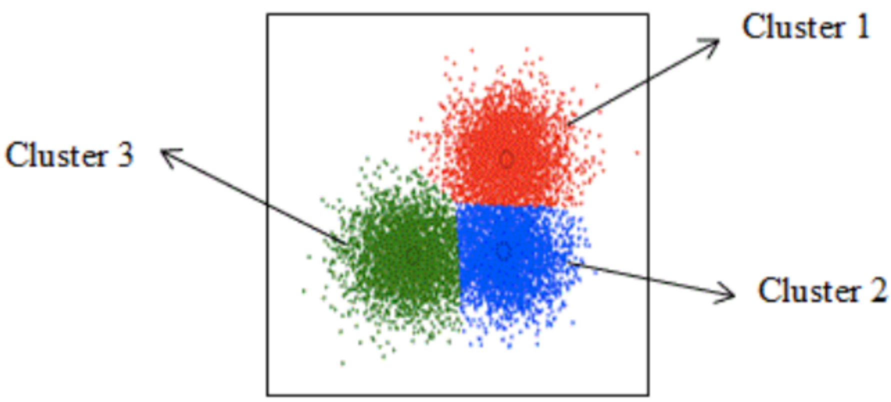


Figure 5.1: Three Clusters of given [70].

an image. A keyword represents different attributes of image. Each feature has a specific value, one cluster made by similar keywords with same values. Relevance feedback algorithm is a keyword based techniques.

Content based clustering represents every information taken from image such as text or shape. The algorithms and tools, which applies statistical formulas, or pattern recognition or etc called content-based clustering method. K-Means clustering which uses pattern recognition to classify images is the most popular example technique[71]. Also, Clustering method can be divided into

- **supervised-clustering** that clustering region can be defined manually
- **unsupervised-clustering** that clustering region already defined automatically.

There are numerous clustering techniques have been used in image segmentation.

5.1 K-Means Clustering

K-Means algorithm is a very popular method used to segment the image to K clusters. Had been invented by Macqueen in 1967, it is unsupervised algorithm that can overcome the typical clustering problem in image segmentation.

The process consists of steps to divide a given image data into a definite number of clusters. The key idea is to assign K centroids to each cluster. Since different locations cause different results, so then it would better to place centroids as much as possible away from each other. Former step is to get points one by one belonging to data set and put it in a nearest centroid. This step is completed successfully until no more point has been left. As a result, an early group age emerges out. In this moment only thing is to recalculate K new centroids as a bar centers of clusters concluded from first process [72].

When K new centroids appear the newer calculation has to be done among previous data set points and other new nearest centroid. As this loop persists on, the K centroids change their places step by step till no more changes have been seen Employing K-Means clustering algorithm may illustrate a certain number of non hierarchical or flat and disjoint clusters [72]. Though, it is appropriate to generate global clusters since it is an unsupervised, non-deterministic, numerical and iterative method. In another word, data vectors in K-Means method are divided into predefined and known number of clusters [73, 74].

At the beginning the centroids (mean) of the predefined clusters are initialized randomly. The dimensions of the centroids are same as the dimension of the data vectors. Each pixel is assigned to the cluster based on the closeness, which is calculated by the Euclidian distance measure when all pixels are clustered, the mean of each cluster is recalculated [75]. This process is repeated until no significant changes result for each cluster mean or for some fixed number of iterations. K-Means algorithm typically consists of following steps:

1. Choosing the k number of clustering manually or randomly:

$$v_i, i = 1, 2, ..., k$$

2. Generating k clusters by assigning each point x_j to nearest cluster mean v_i using Euclidean distance measurement:

$$d_{ij} = ||x_j - v_i||$$

3. Where $X = x_1, x_2, ..., x_n$, is input data points. Compute matrix U corresponds to classification of given points with the binary membership value of j^{th} point to i^{th} cluster in such a way that $U = [u_{ij}]$ where $u_{ij} \in 0, 1$ for all i and j:

$$\begin{Bmatrix} \sum_{j=1}^{k} u_{ij} = 1; & forallj \\ 0 < \sum_{j=1}^{k} u_{ij} < n & forallj \end{Bmatrix}$$

4. Resuming calculation of cluster centers by averaging all pixels in cluster.

$$v_i = \frac{\sum_{j=1}^{n} u_{ij} x_j}{\sum_{j=1}^{n} u_{ij}}; foralli$$

5. If a cluster mean is the same with previous iteration, stop otherwise go to step.

Actually K-Means tries to find one mean in each cluster. It defines means by taking K samples randomly and then follows the two iterations. It assigns each point to nearest mean and substantially moves mean to center of its clusters. The process easily can be seen in Figure 3.4, there are two clusters defining by red and blue colors and for each cluster you can see corresponding centers. This algorithm tries to minimizing an objective function, e.g. a squared error function. The objective function $J(U, V)$ is:

$$J(U, V) = \sum_{j=1}^{k} \sum_{j=1}^{n} \|x_j - v_i\|^2$$

Where $\|x_j - v_i\|$ is a chosen distance measure between a x_j data point and the cluster center, v_i , is an indicator of the distance of the n data points from their respective cluster centers. Here it shows sample Lena image segmented by K-Means is shown as follows:

Figure 5.2: a) Lena image, b) Segmented image using K-Means [75].

To understand better there is a block diagram below, which indicates the K-Means clustering process as well.

Briefly we can said K-Means Clustering follow these step:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Allocate each object to the group that has the closest centroid.
3. When all objects have been allocate, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

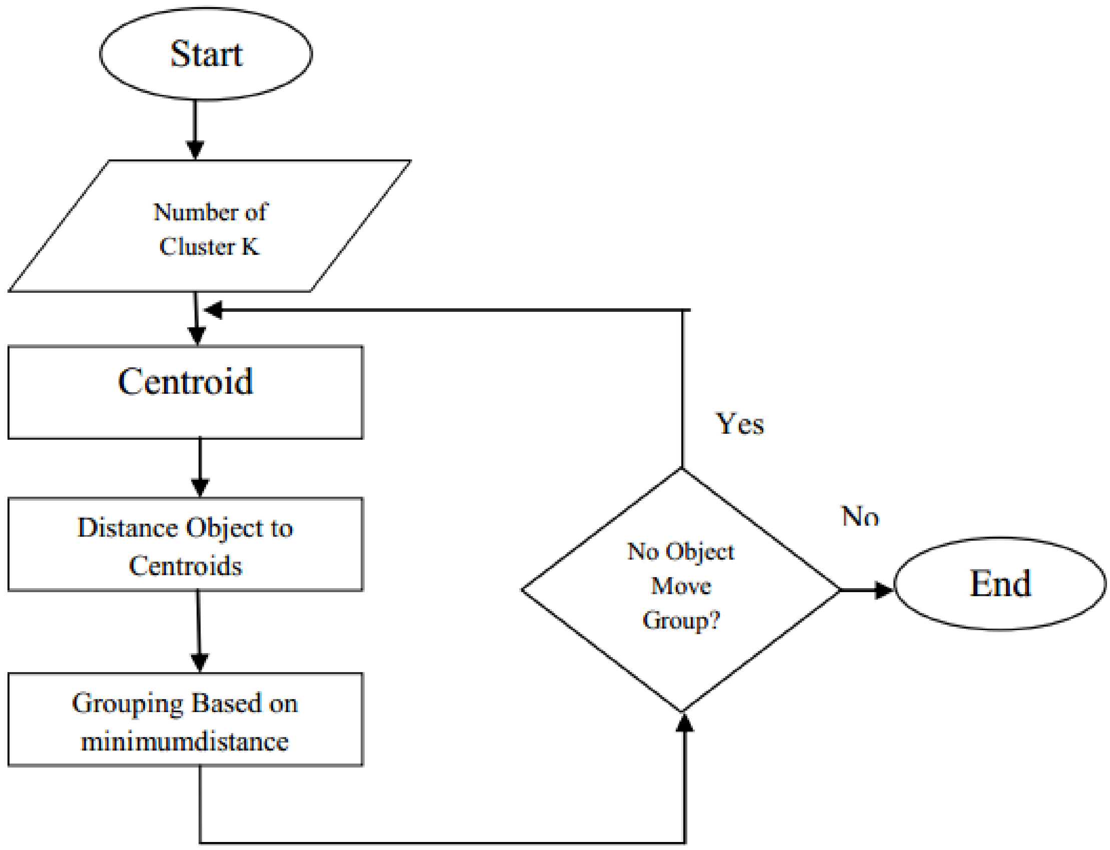


Figure 5.3: Block Diagram of K-Means Process.

The last figure in this part is an illustration of K-Means algorithm. First we choose two cluster centers shown in red color, then the algorithm allocate each objects to centers which is most similarity of them. In next step K-Means updates the cluster means then allocate objects to most similar centers again and this step is repeated until the centroids do not move anymore.

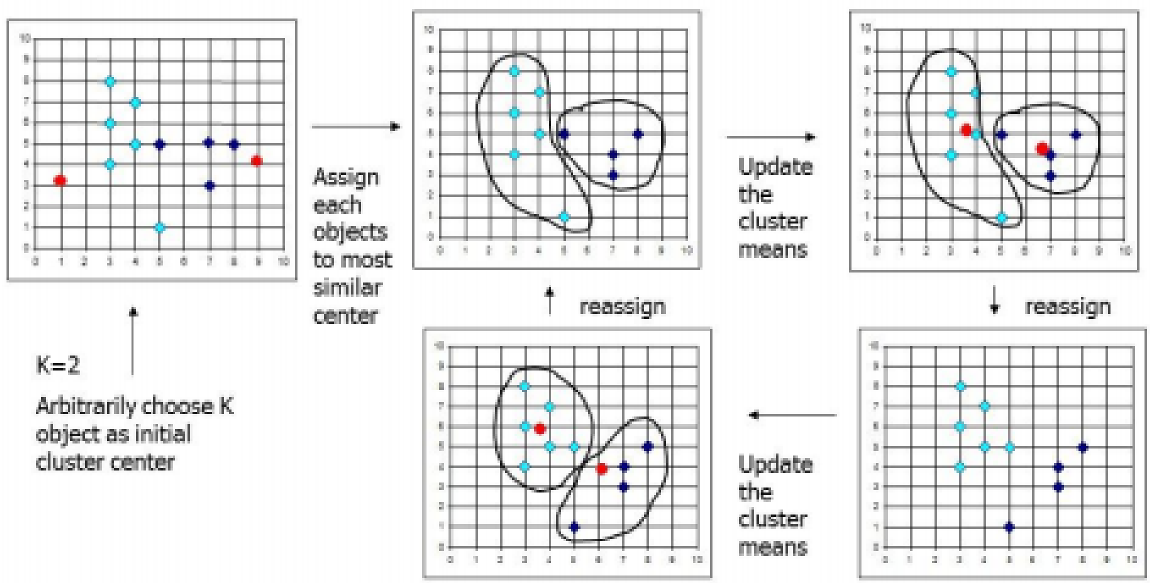


Figure 5.4: Typical K-Means algorithm [76]

Chapter 6

Method

Since this is an Method chapter, the main method of the project explained completely. Moreover, all steps that checked during the project mention by results to show all what has been done and check during this project.

Our method consists of Three major parts:

- Construction of the pattern block and Classified energy(data base)
- Encoding Process ,
- Decoding Process, This part has been done by 2 different step.

6.1 Construction of the Classified Pattern and Energy Blocks

Here we choose our data set which include 5 images. In order to find database. We obtain energy and pattern blocks of each image, collect all results in one matrix, then by using K-means, clustered our results and eliminate similar ones. By this way, we made our data base as 2 matrix:

- pattern block,
- Classified energy.

Let Image data $Im(m,n)$ which m,n that in our case $m=n=256$ and the range of entries are between 0 to 255 that changes to the real values by 0 to 1 where m and n are row and column pixel indices of the whole image[77].

we divided Training images to image blocks by size of $i=j=16,8,4,2$. The pixel location of the kth row and lth column of the block, $B_{r,c}$ is represented by $P_{B_{r,c},kl}$, where the pixel indices are $k=1$ to i and $l=1$ to j. The total number for all images will be $N_B=(M\times N)/(I\times J)$. Moreover, in $B_{r,c}$, r,c represented in the range of M/i and N/j.
All image blocks $B_{r,c}$ from left to the right direction are reshaped as column vectors and constructed The B_{IM} matrix. After the blocking process the image matrix can be written as follows:

$$Im=\begin{bmatrix} B_{1,1} & B_{1,2} & ... & B_{1,(N/j)-1} & B_{1,(N/j)} \\ B_{2,1} & B_{2,2} & ... & B_{2,(N/j)-1} & B_{2,(N/j)} \\ ... & ... & ... & ... & ... \\ B_{(M/i)-1,1} & B_{(M/i)-1,2} & ... & B_{(M/i)-1,(N/j)-1} & B_{(M/i)-1,(N/j)} \\ B_{(M/i),1} & B_{(M/i),2} & ... & B_{(M/i),(N/j)-1} & B_{(M/i),(N/j)} \end{bmatrix} \tag{6.1}$$

B_{Im} is the transformed type of I_M which column vectors are the block of the matrix.

$$B_{Im}=\begin{bmatrix} B_{1,1} & ... & B_{1,(N/j)} & B_{2,1} & ... & ... & B_{(M/i),(N/j)} \end{bmatrix} \tag{6.2}$$

The columns of the matrix B_{Im} are called image block vector (IBV) and the length of the IBV is represented by $L_{IBV}=i\times j(8\times 8=64 or 16\times 16=256, etc.)$ Here we proposed that any ith IBV of lenght $L_{I_{BV}}$ approximated by:
IBV$_i$ =S$_i P_{IP} E_{IE}, (i=1,...,N_B)$ where S_i is a real constant. $IP\in\{1,2,...,N_{IP}\}$

and $IE \in \{1, 2, ..., N_{IE}\}$ are the index number of CPB and CEB and N_{IP} and N_{IE} are the total number of CPB and CEB indices. IP, IE, N_{IP}, N_{IE} are intigers.

$E_{IE}^T = \begin{bmatrix} e_{IE1} & e_{IE2} & ... & e_{IEL_{IBV}} \end{bmatrix}$ is the vector form of CEB that generated utilizing the luminance information of the images and it contains basically the energy characteristics of IBV_i.

Moreover, $S_i E_{IE}$ carries almost maximum energy of IBV_i in the Least Mean Square (LMS). S_i is to scale the luminance level of the IBV_i. P_{IP} is $(L_{IBV} \times L_{IBV})$ diagonal matrix such that

$$P_{IP} = diag \begin{bmatrix} P_{IP1} & P_{IP2} & ... & P_{IPL_{IBV}} \end{bmatrix}. \tag{6.3}$$

P_{IP} is likes a pattern term on the quantity $S_i E_{IE}$ that reflect the distinctive properties of the image block data. It is well known that each IBV can be spanned in a vector space formed by the orthonormal vectors ψ_{ik}. Let the real orthonormal vectors be the columns of a transposed transformation matrix (ψ_i^T)

$$\psi_i^T = \begin{bmatrix} \psi_{i1} & \psi_{i2} & ... & \psi_{iL_{IBV}} \end{bmatrix}. \tag{6.4}$$

It is evident that:

$$IBV_i = \psi_i^T G_i \tag{6.5}$$

where

$$S_i^T = \begin{bmatrix} s_1 & s_2 & ... & s_{L_{IBV}} \end{bmatrix}. \tag{6.6}$$

from the property of $\psi_i^T = \psi_i^{-1}$, the equations $\psi_i IBV_i = \psi_i \psi_i^{-1} S_i$ and $S_i = psi_i IBV_i$ can be obtained respectively. So, IBV_i can be written as a weighted sum of these orthonormal vectors

$$IBV_i = \sum_{k=1}^{L_{IBV}} s_k \psi_{ik}, k = 1, 2, 3, ..., L_{IBV}. \tag{6.7}$$

From the above equation, the coefficients of the $IBVs$ can be obtained as

$$s_k = \psi_{ik}^T IBV_i, k = 1, 2, 3, ..., L_{IBV}. \tag{6.8}$$

let

$$IBV_{it} = \sum_{k=1}^{t} s_k \psi_{ik},$$

be the truncated version of IBV_i such that $1 \leq t \leq L_{IBV}$. it's noted that if $t = L_{IBV}$, then IBV_i will be equal to IBV_{it}. By this way, the approximation error (ϵ_t) is given by:

$$\epsilon_t = IBV_i - IBV_{it} = \sum_{k=t+1}^{L_{IBV}} s_k \psi_{ik}, \tag{6.9}$$

ψ_{ik} are determined by minimizing the expected value of the error vector with respect to ψ_{ik} in the LMS sense. The above-mentioned LMS process results in the following eigenvalue problem [78]. ψ_{ik} computed as the eigenvectors of the correlation matrix (R_i) of the IBV_i. By using orthonormality condition, the LMS error is given by

$$\epsilon_t \epsilon_t^T = \sum_{k=t+1}^{L_{IBV}} s_k^2. \tag{6.10}$$

Let γ_t designate the expected value of the total squared error $\epsilon_t \epsilon_t^T$. Then,

$$\gamma_t = E[\epsilon_t \epsilon_t^T] = \sum_{k=t+1}^{L_{IBV}} E[s_k^2], \tag{6.11}$$

$$E[s_k^2] = E[\psi_{ik}^T (IBV_i^T IBV_i)\psi_{ik}] = \psi_{ik}^T R_i \psi_{ik}, \tag{6.12}$$

where, $R_i = E[IBV_i^T IBV_i]$ is defined as the correlation matrix of IBV_i. In order to obtain the optimum transform, it is desired to find ψ_{ik} that minimizes γ_t for given t, subject to the orthonormality constraint. Using Lagrange multipliers λ_k, we minimize γ_t by taking the gradient of the equation obtained above with respect to ψ_{ik}:

$$\gamma_t = \sum_{k=t+1}^{L_{IBV}} [\psi_{ik}^T R_i \psi_{ik} - \lambda_k(\psi_{ik}^T R_i \psi_{ik} - 1)], \frac{\delta\gamma_t}{\delta\psi_{ik}} = \frac{\delta}{\delta\psi_{ik}} \tag{6.13}$$

$$E[s_k^2] = E[\psi_{ik}^T (IBV_i^T IBV_i)\psi_{ik}] = \psi_{ik}^T R_i \psi_{ik} [\sum_{k=t+1}^{L_{IBV}} [\psi_{ik}^T R_i \psi_{ik} - \lambda_k(\psi_{ik}^T R_i \psi_{ik} - 1)]] = 0, \tag{6.14}$$

$$2R_i\psi_{ik} - 2\lambda_k\psi_{ik} = 0, R_i\psi_{ik} = \lambda_k\psi_{ik}, \tag{6.15}$$

R_i is the correlation matrix. It is real, symmetric with respect to its diagonal elements, positive semidefinite, and Toeplitz matrix [79]

$$R_i = \begin{bmatrix} r_i(1) & r_i(2) & r_i(3) & \dots & r_i(L_{IBV}) \\ r_i(2) & r_i(1) & r_i(2) & \dots & r_i(L_{IBV}-1) \\ r_i(3) & r_i(2) & r_i(1) & \dots & r_i(L_{IBV}-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_i(L_{IBV}) & r_i(L_{IBV}-1) & r_i(L_{IBV}-2) & \dots & r_i(1) \end{bmatrix} \tag{6.16}$$

Obviously, λ_{ik} and ψ_{ik} are the eigenvalues and eigenvectors of the eigenvalue problem under consideration. It is well known that the eigenvalues of R_i are also real, distinct, and non-negative. Moreover, the eigenvectors ψ_{ik} of the R_i

are all orthonormal. Let eigenvalues be sorted in descending order such that $(\lambda_{1i} \geq \lambda_{2i} \geq \lambda_{3i} \geq \ldots._{LIBVi})$ with corresponding eigenvectors. The total energy of the IBV_i is the given by $IBV_{iT} IBV_i$:

$$IBV_{iT} IBV_i = \sum_{k=1}^{L_{IBV}} s_{ik}^{2} = \sum_{k=1}^{L_{IBV}} \lambda_{ik}. \tag{6.17}$$

Equation (15) may be truncated by taking the first $P = 1$ principal components, which have the highest energy of the IBV_i such that:

$$IBV_i \cong \sum_{k=1}^{\rho} s_k \psi_{ik}. \tag{6.18}$$

The simplest form of (16) can be obtained by setting $\rho = 1$. The eigenvector ψ_{ik} is called energy vector, which has the highest energy in LMS sense, may approximate each image block belonging to the IBV_i. Thus,

$\mathrm{IBV}_i \cong s_1 \psi_{i1}$. (6.19)

In this case, one can vary the L_{IBV} LIBV as a parameter in such way that almost all the energy is captured within the first term of (16) and the rest becomes negligible. That is why ψ_{i1} is called the energy vector since it contains most of the useful information of the original IBV under consideration. Once (17) is obtained, it can be converted to an equality by means of a pattern term P_i which is a diagonal matrix for each IBV . Thus IBV_i is computed as:

$\mathrm{IBV}_i = S_i P_i \psi_{i1}$.(6.20)

In (18), diagonal entries p_{ir} of the matrix P_i are determined in terms of the entries of ψ_{i1r} of the energy vector ψ_{i1} and the entries (pixels) IBV_{ir} of the IBV_i by simple devision. Hence,

$$P_{ir} = \frac{IBV_{ir}}{S_i \psi_{i1r}}, (r = 1, 2, ..., L_{IBV}). \tag{6.21}$$

In essence, the quantities p_{ir} of (19) somewhat absorb the energy of the terms eliminated by truncation of (16). In this paper, several tens of thousands of IBVs were investigated and several thousands of energy and pattern blocks were generated. It was observed that the energy and the pattern blocks exhibit repetitive similarities. In this case, one can eliminate the similar energy and pattern blocks and thus, constitute the so-called classified energy and classified pattern block sets with one of a kind or unique blocks. For the elimination process Pearson's correlation coefficient (PCC) [?] is utilized. PCC is designated by ρ_{XY} and given as

$$\rho_{XY} = \frac{\sum_{i=1}^{L}(x_i y_i) - [\sum_{i=1}^{L} x_i \sum_{i=1}^{L} y_i]/L}{\sqrt{[\sum_{i=1}^{L} x_i^2 - (\sum_{i=1}^{L} x_i)^2/L].[\sum_{i=1}^{L} y_i^2 - (\sum_{i=1}^{L} y_i)^2/L]}}. \tag{6.22}$$

In (20)

$$X = \begin{bmatrix} x_1 & x_2 & ... & x_L \end{bmatrix}.$$

and

$$Y = \begin{bmatrix} y_1 & y_2 & ... & y_L \end{bmatrix}.$$

are two sequences subject to comparison, where L is the length of the sequences. It is assumed that the two sequences $0.9 \leq \rho_{xy} \leq 1$. Hence, similar energy and pattern blocks are eliminated accordingly. During the execution of the elimination stage, it is observed that similarity rate of the energy blocks are much higher than the pattern blocks. Because of huge differences in the similarity rate or in other words elimination rate, the numbers of classified energy blocks in the are almost identical if Classified Energy Pattern Block are very limited. This is natural because energy blocks reflect the luminance information of the image blocks, while pattern blocks carry the pattern or variable information in the image blocks. This is in reality related to tasks of these blocks in the method as explained in the beginning of this section. In order to Elimination we use K-Means that we spoke about the procedure in chapter five. In order to clustering, for CEB and CPB we choose 8 Cluster for CPB and 512 Cluster for CEB. These quantity Has been chosen based on last step we have been done in 6.22. We put all Signature values of all Images beside and made main Signature value matrix. The same procedure has been done for Energy blocks too. By this way we have 2 big matrix by sizes that you see in table blow. then we use K-Means and we made our Database CEB and CPB.

These representative energy and pattern blocks are renamed as classified energy and pattern blocks and constitute the CEPB. Thus, the energy blocks which have unique shapes are combined under the set called classified energy block $CEB = E_{nie}; n_{ie} = 1, 2, 3, ..., N_{IE}$ set.The integer N_{ie} designates the total number of elements in this set. Similarly, reduced pattern blocks are combined under the set called classified pattern block $CPB = E_{nip}; n_{ip} = 1, 2, 3, ..., N_{IP}$ set. The N_{ip} designates the total number of unique pattern sequences in CPB set. Some similar energy and pattern blocks are depicted in Figures5and6,respectively. Computational steps and the details of the encoding and decoding algorithms are given in next sections respectively.

Table 6.1: Energy and signature values metrics size

Block sizes	2 ×2	4 ×4	8 ×8
E_{IE}	4 ×1638	16 ×4096	64 ×1024
S_i	4 ×16384	16 ×4096	64 ×1024
P_{IP}	4 ×16384	16 ×4096	64 ×1024

As table shows by increasing the block size the number of samples decrease for both E_{IE} and P_{IP}.

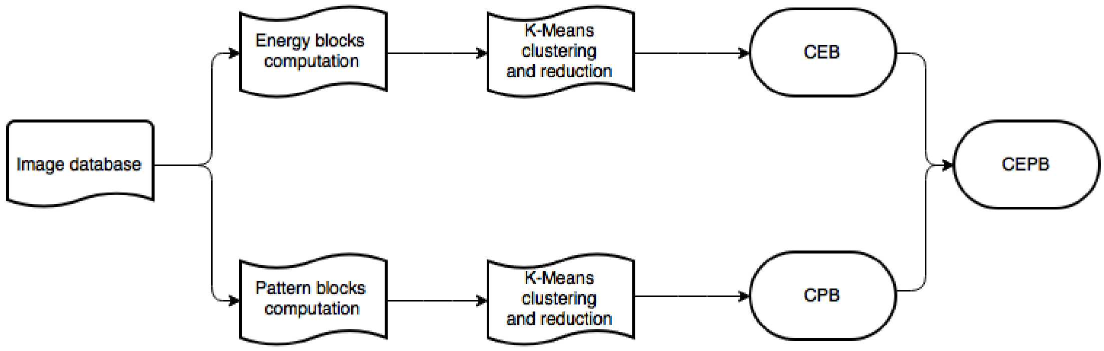


Figure 6.1: CEPB construction

Table 6.2: CPB and CEB matrices sizes for different frame lenght

Block Sizes	2 ×2	4 ×4	8 ×8
CPB	30 ×4	30 ×16	30 ×64
CEB	256 ×4	256 ×16	256 ×64

6.2 Encoding Based on Fixed Block Sizes Method

First of all we defined input images' Energy and Pattern blocks. All these procedure are same as the procedure that we did in Construction of the Classified Pattern and Energy Blocks section to find training Images signature and energy blocks. Then, by using the minimum distance or the total error $\delta_{I\tilde{E}} = \|IBV_i - G_{I\tilde{E}}E_{I\tilde{E}}\|^2$ for all $I\tilde{E} = 1, 2, 3, ..., IE, ..., N_{IE}$ the appropriate E_{IE} founded from CEB which yeilds the index IE of the E_{IE}. So, $\delta_{I\tilde{E}} = min\|IBV_i - S_{I\tilde{E}}E_{I\tilde{E}}\|^2 = \|IBV_i - S_{IE}E_{IE}\|^2$. By storing the index number of IE that refers to E$_{IE}$,IBV$_i \approx S_{IE}E_{IE}$. All steps that done to find the appropriate E_{IE} founded from CEB should be done to find appropriate P_{IP} founded from CPB. So, by minimizing the error $I - \tilde{P} = 1, 2, 3, ..., Ip, ..., N_{IP}$ that yields the index IP of P_{IP},

$$\delta_{I\tilde{P}} = min\|IBV_i - S_{IE}P_{I\tilde{P}}E_{IE}\|^2 = \|IBV_i - S_{IE}P_{IP}E_{IE}\|^2. \qquad (6.23)$$

Then, storing the index number IP that refers to P_{IP}. At the end of this part, the best E_{IE} and P_{IP} founded by appropriate selection. So, IBV_i is best described in terms of the patterns of E_{IE} and P_{IP} as $IBV_i \cong S_{IE}P_{IE}E_{IE}$. In order to compute new block scaling coefficient S_{IE}, what done before for training test by repeated as $S_i = (P_{IP}E_{IE})^T IBV_i/(P_{IP}E_{IE})^T(P_{IP}E_{IE})$ using fixed E_{IE} and P_{IP}, to further minimize the distance between the vectors IBV_i and $S_{IE}P_{IP}E_{IE}$ in the LMS sense. In this case, the global minimum of the error is obtained and it is given by $\delta_{Global} = \|IBV_i - S_iP_{IP}E_{IE}\|^2$. At this step $IBV_{Ai} = S_iP_{IP}E_{IE}$. At this step we have data base that conclude all images $2 \times 2, 4 \times 4, 8 \times 8$ blocks Energy and pattern blocks. and the classification result based on all these images for both Energy and pattern blocks.

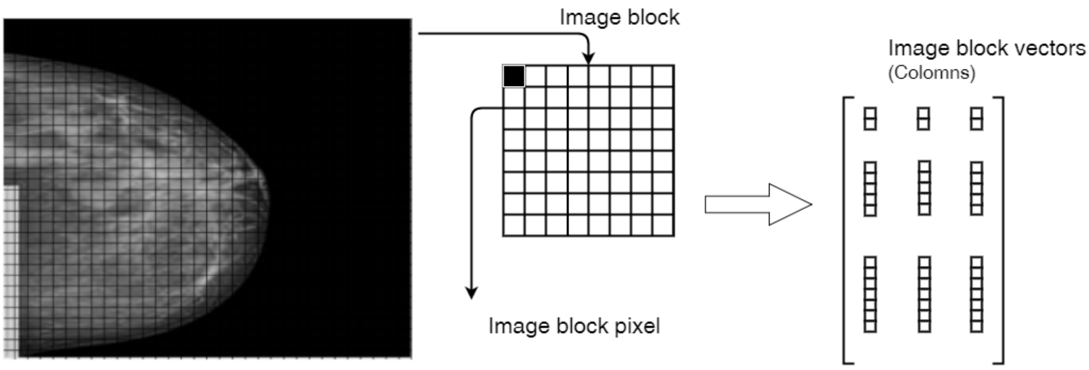


Figure 6.2: Partitioning of an image into the image blocks and reshaping as vector form,(Image block size:$(i \times j)$, Image block pixel:$P_{B_{r,c},k,l}$, Image block vector: IBV_i by size: $(i \times j) \times N_B$)

6.3 Encoding Based on Quadtree Method

In this part encoding has been done based on Quadtree.At first part images blocks have been found. In this regard we choose different thresholds by quantity of2,5,10,15,20,30. Moreover, we made blocking by different size of block [80]:

- min block size=2, max block size=16
- min block size=4, max block size=8
- min block size=2, max block size=8

All other step step is same as Encoding section. the only difference is to do it by different block size.

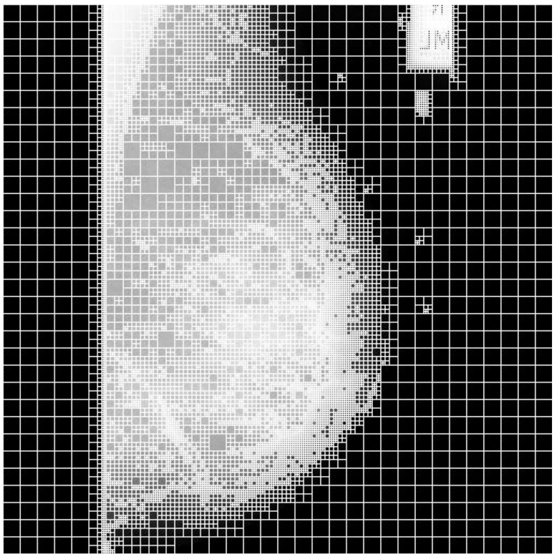


Figure 6.3: Blocked Image by Quadtree method, max block size=16, min block size=2

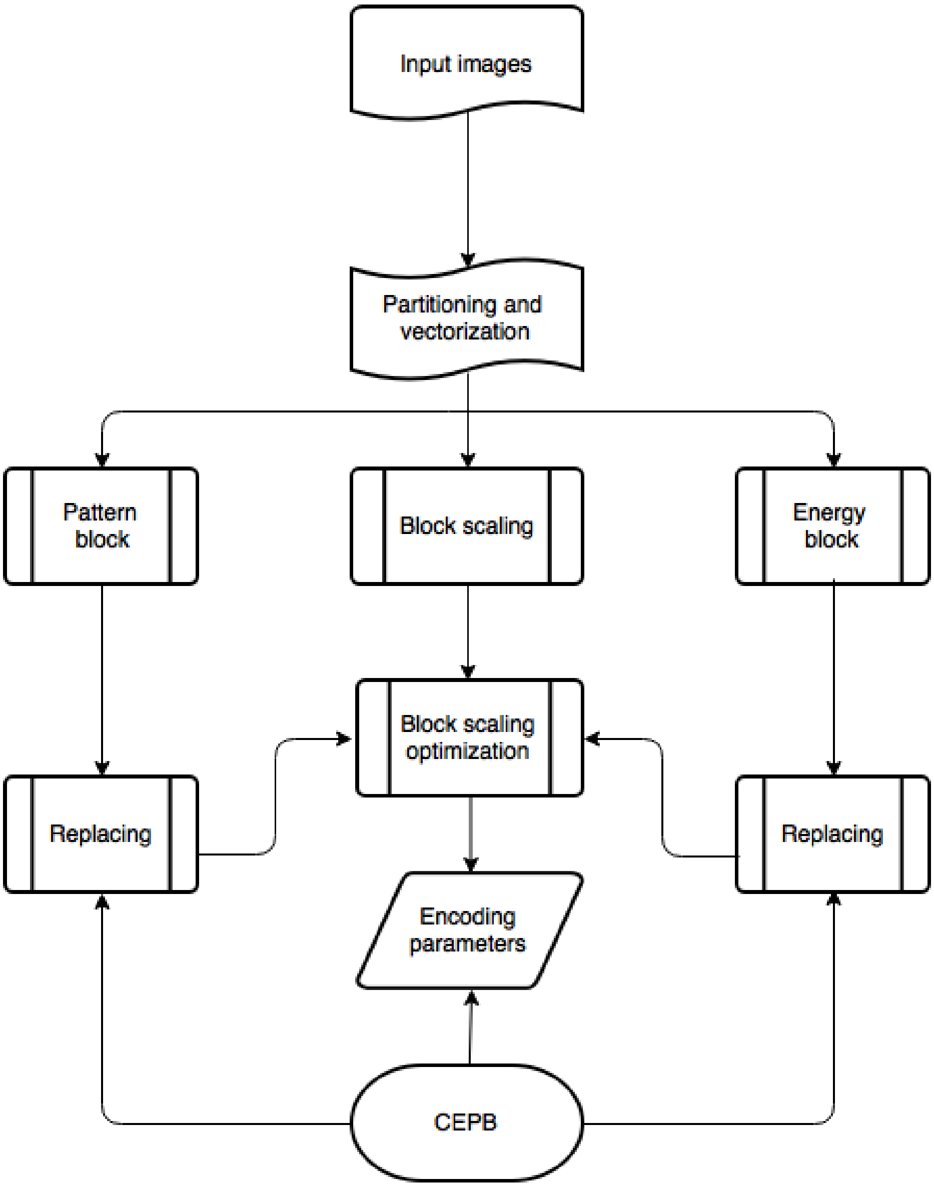


Figure 6.4: Encoding

6.4 Decoding Based on Fixed Block Sizes Method

As we mentioned in beginning of chapter, we have 2 parts in this step.

- Decoding Based on fixed block sizes Method,
- Decoding based on Quadtree Method.

There are three types of inputs for decoding:

- encoding parameters S_i, IP and IE which best represent the corresponding image block vector IBV_i of the input image received from transmitter part for each image block vector of input images.

- Size of IBV_i of the $Im(m,n)(L_{IBV} = i \times j fori = j = [2,4,8,16]$;

- The $CEPB(CEB = E_{IE} = 1,2,...N_{IE} and CPBP_{IP}; IP = 1,2,...,N_{IP})$ that located in receiver part.

  In order to decoding computational steps, after receiving encoding parameters S_i, IP and IE of IBV_i from the transmitter, the corresponding IEth and IPth classified Energy and pattern blocks are pulled from the CEPB. In the next step, using the mathematical model $IBV_{Ai} = S_i P_{IP} E_{IE}$ approximated image block vector IBV_i is constructed. By repeating this step for each IBV approximated version of $(\hat{B_{IM}})$ of the B_{IM} is generated:

$$\hat{B_{Im}} = \begin{bmatrix} \hat{B}_{1,1} & \dots & \hat{B}_{1,(N/j)} & \hat{B}_{2,1} & \dots & \dots & \hat{B}_{(M/j),(N/j)} \end{bmatrix} \tag{6.24}$$

- Decoded version of original image made by reshaping $(\hat{B_{IM}})$ as :

$$\hat{Im} = \begin{bmatrix} \hat{B}_{1,1} & \hat{B}_{1,2} & \dots & \hat{B}_{1,(N/j)-1} & \hat{B}_{1,(N/j)} \\ \hat{B}_{2,1} & \hat{B}_{2,2} & \dots & \hat{B}_{2,(N/j)-1} & \hat{B}_{2,(N/j)} \\ \dots & \dots & \dots & \dots & \dots \\ \hat{B}_{(M/j)-1,1} & \hat{B}_{(M/j)-1,2} & \dots & \hat{B}_{(M/j)-1,(N/j)-1} & \hat{B}_{(M/j)-1,(N/j)} \\ \hat{B}_{(M/j),1} & \hat{B}_{(M/j),2} & \dots & \hat{B}_{(M/j),(N/j)-1} & \hat{B}_{(M/j),(N/j)} \end{bmatrix} \tag{6.25}$$

At low bit rates blocking effect has been shown itself. Especially, when the size of the CEPB is highly reduced or the size of the image blocks are increased from 8×8 to 16×16.

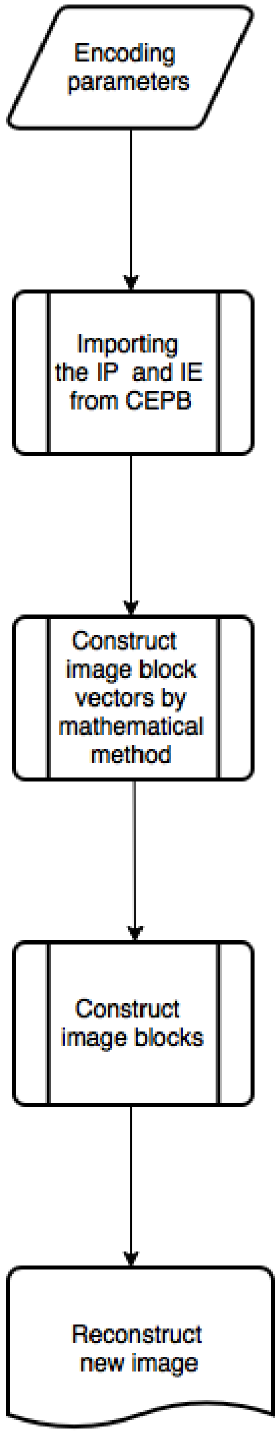


Figure 6.5: Decoding

6.5 Decoding based on Quadtree Method

In this part we used Quadtree method based on different block size from 2×2 to 16×16 block sizes. First of all, after pass image from Quadtree and make blocking, as what we do in Decoding Based on one size block part, replace each block by the most similar blocks in our database. In fact, all parts have been done done here is same as first method, but not only for one size, but also for different sizes from 2×2 to 16×16. blow results for different Quadtree method are shown.

Chapter 7

Experimental Results and Discussion

7.1 Data Sets

5 gray-scale, 8bits/pixel, 256×256 images has been used as data sets.

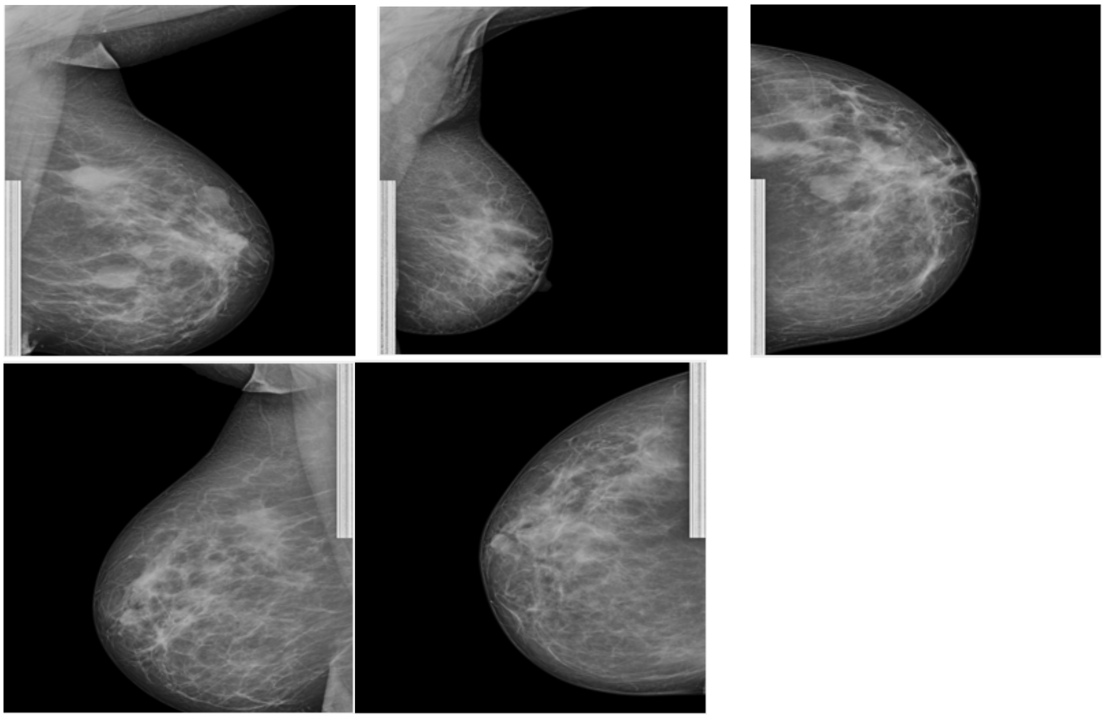


Figure 7.1: Data set images,8bits/pixel, Size: 256×256

7.2 Evaluation Metrics

Objective image and video quality metrics such as peak signal-to-noise ratio (PSNR) and mean squared error (MSE) are the most widely used objective image quality/distortion metrics and they can predict perceived image and video quality automatically. It should be also noted that these metrics are also criticized because they are not correlating well with the perceived quality measurement. Recently, image and video quality assessment research is trying to develop new objective image and video quality measures such as structural-similarity-based image quality assessment (SSIM) by considering HVS characteristics. Almost all the works in the literature consider the PSNR and MSE as an evaluation metrics to measure the quality of the image. Therefore, as a starting point at least for the comparisons, the performance of the newly proposed method is measured using PSNR and MSE metrics.

7.2.1 Peak Signal-to-Noise Ratio (PSNR)

PSNR is the ratio between the signals maximum power and the power of the signals noise. The higher PSNR means better quality of the reconstructed image. The PSNR can be computed as

$$PSNR = 20log_{10}\frac{b}{\sqrt{MSE}} \tag{7.1}$$

where is the largest possible value of the image signal (typically 255 or 1). The PSNR is given in decibel units (dB) [81].

7.2.2 Mean Squared Error (MSE)

MSE represents the cumulative squared error between the original and the reconstructed image [82], whereas PSNR represents a measure of the peak error. The MSE can be described as the mean of the square of the differences in the pixel values between the corresponding pixels of the two images. MSE can be written as

$$MSE = \frac{1}{MN} \sum_{M}^{i=1} \sum_{N}^{j=1} [Im(m,n) - \hat{I}m(m,n)]^2 \tag{7.2}$$

where $Im(m,n)$ and $\hat{I}m(m,n)$ are the original and the reconstructed images, respectively.$M \times N$ is the dimension of the images [83]. In our experiments the dimension of the images is $M = N = 256$.

7.2.3 Compression Ratio (CR)

CR is defined as the ratio of the total number of bits required to represent the original and reconstructed image blocks[84]. Other representation of the CR is the bpp:

$$CR = \frac{bit_{original}}{bit_{reconstructed}}, \tag{7.3}$$

$$bpp(bitperpixel) = \frac{\sqrt{L_{IBV}}}{CR}. \tag{7.4}$$

7.2.4 Structural Similarity (SSIM)

The structural similarity (SSIM) index is a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos [85].

SSIM is used for measuring the similarity between two images. The SSIM index is a full reference metric; in other words, the measurement or prediction of image quality is based on an initial uncompressed or distortion-free image as reference. SSIM is designed to improve on traditional methods such as peak signal-to-noise ratio (PSNR) and mean squared error (MSE), which have proven to be inconsistent with human visual perception.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{7.5}$$

In which:

μ_x equal to average of x,

μ_y equal to average of y,

σ_x^2 equal to variance of x,

$\sigma{-}_x^2$ equal to variance of y,

σ_{xy} equals to covariance of x, y,

$c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator,

L the dynamic range of the pixel-values (typically this is $2^{bitsperpixel} - 1$),

$k_1 = 0.01$ and $k_2 = 0.03$ by default.

The SSIM index satisfies the condition of symmetry:

$$SSIM(x, y) = SSIM(y, x) \tag{7.6}$$

7.2.5 Arithmetic Compression Rate

Arithmetic coding is a common algorithm used in both lossless and lossy data compression algorithms. It is an entropy encoding technique, in which the frequently seen symbols are encoded with fewer bits than lesser seen symbols. It has some advantages over well-known techniques such as Huffman coding. Arithmetic compression rate has been found in this experiment too [86].

7.3 Experimental Results based on Fixed Block size method

The total number needed to represent $n \times n$ block size for each original image is $(n \times n) \times 8bits$.The total number of classified Energy and pattern blocks are determined as CPB and CEB sets like table blow:

It is also concluded that N_{IE} and N_{IP} are represented by 5 bits and 7 bits, respectively. For representation of the block scaling coefficient (BSC) 5 bits are

Table 7.1: Bit allocation table for the experiments

L_{IBV}	$Numberofbits$ $(withoutclustering)$	$CEPBsize$ $(withclustering)$	$Numberofbitsrequired$ $(withclustering)$
		$CEB = 30 < 2^5$	$CEB = 5$
2×2	$(2 \times 2) \times 8$	$CPB = 256 = 2^8$	$CPB = 7$
		$BSC = 5$	$BSC = 5$
		$CEB = 30 < 2^5$	$CEB = 5$
4×4	$(4 \times 4) \times 8$	$CPB = 256 = 2^8$	$CPB = 7$
		$BSC = 5$	$BSC = 5$
		$CEB = 30 < 2^5$	$CEB = 5$
8×8	$(8 \times 8) \times 8$	$CPB = 256 = 2^8$	$CPB = 7$
		$BSC = 5$	$BSC = 5$
		$CEB = 30 < 2^5$	$CEB = 5$
16×16	$(16 \times 16) \times 8$	$CPB = 256 = 2^8$	$CPB = 7$
		$BSC = 5$	$BSC = 5$

good enough. As a result, 17 bits are required in total in order to represent the $n \times n$ blocks of the images. In this case, the compression ratio will be computed as follows:

$$CR = \frac{bit_{original}}{bit_{reconstruct}} = \frac{(n \times n) \times 8}{(5 + 7 + 5)} \tag{7.7}$$

or

$$bpp = \frac{\sqrt{L_{IBV}}}{CR} = \frac{\sqrt{n \times n}}{CR} \tag{7.8}$$

Result for CR values are shown below:

Table 7.2: CR values for different block size

Block size $(i \times j)$	Compression Rate
2×2	$CR = \frac{(2\times2)\times8}{(5+7+5)} = 1,8823$
4×4	$CR = \frac{(4\times4)\times8}{(5+7+5)} = 7,5294$
8×8	$CR = \frac{(8\times8)\times8}{(5+7+5)} = 30,1176$
16×16	$CR = \frac{(16\times16)\times8}{(5+7+5)} = 120,4705$

It is clearly understood from the evaluation results given in the tables that, the performance of the proposed method depends on the size of the L_{IBV}. If the size of the L_{IBV} is increased in order to achieve higher compression ratios or lower bit rates, the performance of the method is getting worse and the blocking effect is also getting visible. Even in this case, it is remarkable that the PSNR levels are not affected dramatically.

Table 7.3: Fixed block size method average results

Block size ($i \times j$)	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
2×2	1,8824	4,8783	47,9622	1,64E-05	0,9925
4×4	7,5294	22,2425	36,4762	2,60E-04	0,9543
8×8	30,1176	61,7489	34,3565	4,59E-04	0,9504
16×16	120,4706	307,3202	30,8865	9,52E-04	0,9019

The same results has been seen in Time results too. By Increasing L_{IBV} time results decrease too.

Table 7.4: Fixed block size method average time results

Block size ($i \times j$)	Compression Rate	encoding time	decoding time	arithmetic encoding time	arithmetic decoding time
2×2	1,8824	117,6781	27,5033	1,6184	1,7993
4×4	7,5294	31,1423	6,9217	0.27959685	0,3100
8×8	30,1176	12,9799	1,7772	0,2008	0,2196
16×16	120,4706	27,2473	0,4337	0,0401	0,0500

Table 7.5: Results for fix block size method, i=j=2

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	1,8824	4,4498	46,9407	2,00E-05	0,9932
mdb012	1,8824	4,9086	46,6928	2,10E-05	0,9931
mdb013	1,8824	4,6671	48,2657	1,50E-05	0,9930
mdb014	1,8824	4,8368	48,1203	1,50E-05	0,9925
mdb015	1,8824	4,4841	48,8046	1,30E-05	0,9938
mdb016	1,8824	4,8916	47,7139	1,70E-05	0,9936
mdb017	1,8824	4,5938	50,6293	9,00E-06	0,9959
mdb018	1,8824	5,0938	49,6178	1,10E-05	0,9952
mdb019	1,8824	5,0101	46,3925	2,30E-05	0,9891
mdb020	1,8824	4,8723	46,5016	2,20E-05	0,9905
mdb021	1,8824	4,8029	46,8190	2,10E-05	0,9916
mdb022	1,8824	4,9905	48,2186	1,50E-05	0,9924
mdb023	1,8824	4,9360	47,8055	1,70E-05	0,9911
mdb024	1,8824	4,9976	48,8608	1,30E-05	0,9927
mdb025	1,8824	4,8547	47,6037	1,70E-05	0,9910
mdb026	1,8824	4,9190	48,3745	1,50E-05	0,9915
mdb027	1,8824	4,9732	48,5859	1,40E-05	0,9924
mdb028	1,8824	5,1108	49,0185	1,30E-05	0,9924
mdb029	1,8824	4,8893	46,0490	2,50E-05	0,9926
mdb030	1,8824	5,2849	48,2289	1,50E-05	0,9924
Average	1,8824	4,8783	47,9622	1,66E-05	0,9925

7.4 Experimental Results Based on Quadtree Method

About Quadtree method for all block sizes all results shown that by increasing Threshold Compression Rates and MSE have increases and SSIM, PSNR and all time results has decreases. the best resolution achieve in lower threshold. However,by decreasing block sizes these changes shown themselves by lower changes. in the 4,8 block size we cannot see these changes in SSIM, PSNR and MSE so visible. but time results affected directly.

As it has been seen in Quadtree results table in this method threshold and block size affect compression rate, PSNR and other parametes.

By increasing the threshold, compression rate,mse increase and other parameters all have decrease. But by increasing the threshold and compression rate

Table 7.6: Results for fixed block size method, i=j=4

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	7,5294	21,0947	33,7373	4,23E-04	0,9574
mdb012	7,5294	22,7043	33,4445	4,52E-04	0,9530
mdb013	7,5294	20,3433	35,4842	2,83E-04	0,9593
mdb014	7,5294	20,6722	36,3591	2,31E-04	0,9600
mdb015	7,5294	20,4114	37,3939	1,82E-04	0,9615
mdb016	7,5294	21,1475	37,1076	1,95E-04	0,9577
mdb017	7,5294	21,4363	40,2669	9,40E-05	0,9748
mdb018	7,5294	22,7733	40,5861	8,70E-05	0,9745
mdb019	7,5294	22,5248	33,6278	4,34E-04	0,9331
mdb020	7,5294	21,5420	35,4231	2,87E-04	0,9453
mdb021	7,5294	22,5772	33,6582	4,31E-04	0,9443
mdb022	7,5294	22,5210	39,5741	1,10E-04	0,9573
mdb023	7,5294	23,0659	34,7530	3,35E-04	0,9494
mdb024	7,5294	23,6699	38,2689	1,49E-04	0,9584
mdb025	7,5294	22,4881	34,7654	3,34E-04	0,9338
mdb026	7,5294	22,3063	37,8655	1,63E-04	0,9431
mdb027	7,5294	23,3536	35,5588	2,78E-04	0,9483
mdb028	7,5294	24,0477	38,5869	1,38E-04	0,9506
mdb029	7,5294	22,7912	33,1074	4,89E-04	0,9595
mdb030	7,5294	23,3786	39,9550	1,01E-04	0,9654
average	7,5294	22,2425	36,4762	2,60E-04	0,9543

blocking effect shown itself more. But by increasing the threshold and compression rate blocking effect shown itself more.

In the comparison compression with out quadtree and with quadtree we can see the average CR for all quadtree results in three group are in the better and higher values and as we check the image we can see differences too.

Table 7.7: Results for fixed block size method, i=j=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	30,1176	59,7411	32,2547	5,95E-04	0,9505
mdb012	30,1176	60,1800	28,8975	1,29E-03	0,9389
mdb013	30,1176	60,3601	35,0293	3,14E-04	0,9589
mdb014	30,1176	58,8823	31,3114	7,39E-04	0,9532
mdb015	30,1176	60,1800	37,9976	1,59E-04	0,9612
mdb016	30,1176	62,0019	35,3103	2,94E-04	0,9559
mdb017	30,1176	66,3992	34,6605	3,42E-04	0,9658
mdb018	30,1176	66,7203	33,1749	4,81E-04	0,9660
mdb019	30,1176	63,0760	34,1677	3,83E-04	0,9336
mdb020	30,1176	59,1747	30,7799	8,36E-04	0,9353
mdb021	30,1176	61,8994	33,6220	4,34E-04	0,9431
mdb022	30,1176	62,1194	40,4755	9,00E-05	0,9597
mdb023	30,1176	62,5194	36,0530	2,48E-04	0,9480
mdb024	30,1176	60,5693	32,4528	5,68E-04	0,9504
mdb025	30,1176	61,7827	36,4375	2,27E-04	0,9363
mdb026	30,1176	60,8364	32,1229	6,13E-04	0,9371
mdb027	30,1176	61,8994	37,3328	1,85E-04	0,9507
mdb028	30,1176	60,6955	33,0948	4,90E-04	0,9454
mdb029	30,1176	61,7536	30,8579	8,21E-04	0,9522
mdb030	30,1176	64,1881	41,0961	7,80E-05	0,9658
average	30,1176	61,7489	34,3565	4,59E-04	0,9504

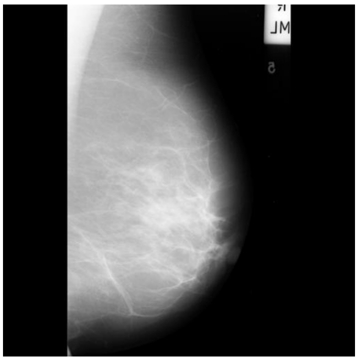


Figure 7.2: Main mdb014 Image, size 256 × 256

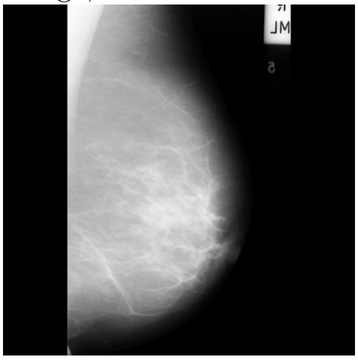


Figure 7.3: mdb014 Reconstruct image by 2 × 2 Block size, CR=1,882353, PSNR=48,12, MSE=0,000015, ssim=0,992611

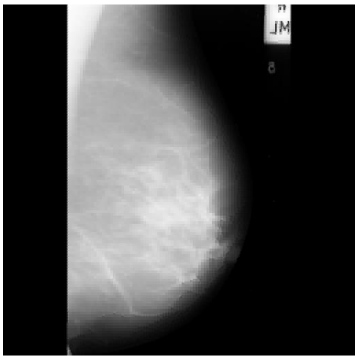


Figure 7.4: mdb014 Reconstruct image by 4 × 4 Block size, CR=7,529412, PSNR=36,3591, MSE=0,000231, ssim=0,960011

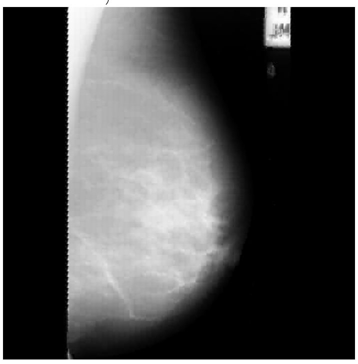


Figure 7.5: mdb014 reconstructed image by 8 × 8 block size, CR=30,117647, PSNR=31,31141, MSE=0,0007395, ssim=0,953234

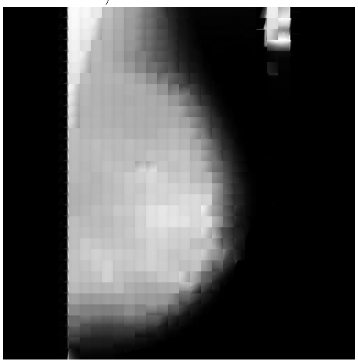


Figure 7.6: mdb014 reconstructed image by 16 × 16 block size, CR=120,470688, PSNR=30,652391, MSE=0,000861, ssim=0,903742

Figure 7.7: Effect of increasing block size

Table 7.8: Results for fixed block size method, i=j=16

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	120,4706	304,1114	27,8752	1,63E-03	0,9073
mdb012	120,4706	310,2296	25,2334	3,00E-03	0,8712
mdb013	120,4706	284,6298	31,0023	7,94E-04	0,9178
mdb014	120,4706	278,2845	30,6524	8,61E-04	0,9037
mdb015	120,4706	295,8736	33,2241	4,76E-04	0,9219
mdb016	120,4706	274,4963	30,7975	8,32E-04	0,9100
mdb017	120,4706	313,1947	31,2327	7,53E-04	0,9284
mdb018	120,4706	328,9134	29,4916	1,12E-03	0,9371
mdb019	120,4706	308,0423	29,2527	1,19E-03	0,8678
mdb020	120,4706	304,8186	29,0864	1,23E-03	0,8754
mdb021	120,4706	313,1947	29,9486	1,01E-03	0,8877
mdb022	120,4706	305,5291	34,1809	3,82E-04	0,9172
mdb023	120,4706	320,0781	31,5340	7,02E-04	0,9020
mdb024	120,4706	317,3656	32,5630	5,54E-04	0,9035
mdb025	120,4706	308,4047	31,4839	7,11E-04	0,8756
mdb026	120,4706	313,1947	32,8330	5,21E-04	0,8816
mdb027	120,4706	314,6987	32,2280	5,99E-04	0,8991
mdb028	120,4706	318,5225	33,2283	4,76E-04	0,8930
mdb029	120,4706	315,4561	27,3541	1,84E-03	0,9074
mdb030	120,4706	317,3656	34,5276	3,53E-04	0,9308
average	120,4706	307,3202	30,8865	9,52E-04	0,9019

Table 7.9: Time results for fixed block size method, i=j=2

name	Compression Rate	encoding time	decoding time	arithmetic encoding time	arithmetic decoding time
mdb011	1,8824	83,8576	24,3184	1,5618	1,6168
mdb012	1,8824	78,6266	23,5174	1,0275	1,1649
mdb013	1,8824	76,6076	24,5558	1,4707	1,5894
mdb014	1,8824	84,1880	22,8023	1,2623	1,3624
mdb015	1,8824	81,2746	23,8571	1,4472	1,5369
mdb016	1,8824	73,2993	22,1082	1,1949	1,2946
mdb017	1,8824	136,5233	29,0306	2,1171	2,3221
mdb018	1,8824	135,4280	29,3489	1,5134	1,7255
mdb019	1,8824	136,4756	28,9085	1,7252	1,9012
mdb020	1,8824	133,0402	29,0540	1,8456	2,0724
mdb021	1,8824	135,3536	32,5237	1,9096	2,1171
mdb022	1,8824	132,3471	29,2352	1,7095	1,8877
mdb023	1,8824	136,7210	31,8326	1,9523	2,2457
mdb024	1,8824	134,9423	28,6632	1,6309	1,8441
mdb025	1,8824	134,3141	28,0721	1,8909	2,0764
mdb026	1,8824	133,2489	27,9078	1,7673	1,9975
mdb027	1,8824	129,9531	27,6964	1,7019	1,9062
mdb028	1,8824	130,4838	29,0794	1,5249	1,7016
mdb029	1,8824	130,6529	27,6636	1,7579	1,9749
mdb030	1,8824	136,2244	29,8897	1,3578	1,6491
average	1,8824	117,6781	27,5033	1,6184	1,7993

Table 7.10: Time results for fixed block size method, i=j=4

name	Compression Rate	encoding time	decoding time	arithmetic encoding time	arithmetic decoding time
mdb011	7,5294	21,6013	6,0131	0,2253	0,2485
mdb012	7,5294	20,0153	5,8585	0,1969	0,2130
mdb013	7,5294	22,4019	6,2543	0,2545	0,2845
mdb014	7,5294	20,9434	6,2082	0,2276	0,2577
mdb015	7,5294	23,1296	6,4475	0,3306	0,3921
mdb016	7,5294	20,9744	6,7492	0,3866	0,4261
mdb017	7,5294	34,6815	8,7043	0,4450	0,3982
mdb018	7,5294	35,0179	7,2491	0,2746	0,3122
mdb019	7,5294	36,9732	7,2798	0,3486	0,3534
mdb020	7,5294	36,8239	7,2281	0,3403	0,3845
mdb021	7,5294	35,7339	7,2254	0,2748	0,3195
mdb022	7,5294	35,7426	6,9433	0,2763	0,3183
mdb023	7,5294	36,5211	7,3479	0,2544	0,2877
mdb024	7,5294	34,3090	6,7337	0,2325	0,2777
mdb025	7,5294	35,4288	6,9408	0,2897	0,3165
mdb026	7,5294	34,5417	6,8924	0,2838	0,3277
mdb027	7,5294	34,9815	6,8414	0,2230	0,2561
mdb028	7,5294	33,5504	6,9443	0,2146	0,2401
mdb029	7,5294	33,9532	7,0073	0,2704	0,3056
mdb030	7,5294	35,5209	7,5657	0,2424	0,2813
average	7,5294	31,1423	6,9217	0,2796	0,3100

Table 7.11: Time results for fixed block size method, i=j=8

name	Compression Rate	encoding time	decoding time	arithmetic encoding time	arithmetic decoding time
mdb011	30,1176	8,6243	1,5220	0,1512	0,1536
mdb012	30,1176	8,6616	1,5184	0,1551	0,1352
mdb013	30,1176	9,1110	1,5794	0,1449	0,1468
mdb014	30,1176	9,5278	1,4623	0,1770	0,1772
mdb015	30,1176	19,8836	1,4851	0,1329	0,1321
mdb016	30,1176	13,3756	1,9323	0,2266	0,2457
mdb017	30,1176	13,7331	1,8772	0,1880	0,2120
mdb018	30,1176	13,7748	1,8579	0,1832	0,2072
mdb019	30,1176	13,6800	2,2159	0,2213	0,2783
mdb020	30,1176	13,4222	1,7976	0,2292	0,2571
mdb021	30,1176	13,4803	1,8300	0,2205	0,2399
mdb022	30,1176	13,4524	1,7379	0,2097	0,2343
mdb023	30,1176	13,6269	1,8078	0,2094	0,2350
mdb024	30,1176	14,4380	1,7400	0,2265	0,2546
mdb025	30,1176	12,9640	1,7484	0,2121	0,2342
mdb026	30,1176	13,5919	1,7184	0,2170	0,2409
mdb027	30,1176	13,2581	1,9861	0,2427	0,2738
mdb028	30,1176	13,2634	1,7688	0,2220	0,2439
mdb029	30,1176	13,4043	2,0322	0,2198	0,2518
mdb030	30,1176	14,3238	1,9256	0,2269	0,2382
average	30,1176	12,9799	1,7772	0,2008	0,2196

Table 7.12: Time results for fixed block size method, i=j=16

name	Compression Rate	encoding time	decoding time	arithmetic encoding time	arithmetic decoding time
mdb011	120,4706	10,6693	0,3784	0,0251	0,0283
mdb012	120,4706	10,3077	0,3676	0,0206	0,0229
mdb013	120,4706	11,1375	0,3715	0,0434	0,0275
mdb014	120,4706	11,6281	0,3758	0,0320	0,0289
mdb015	120,4706	12,7660	0,3336	0,0322	0,0355
mdb016	120,4706	35,6555	0,4726	0,0612	0,0715
mdb017	120,4706	32,6552	0,4635	0,0361	0,0561
mdb018	120,4706	32,3935	0,4624	0,0369	0,0506
mdb019	120,4706	33,3807	0,4676	0,0529	0,0604
mdb020	120,4706	32,6370	0,4524	0,0389	0,0600
mdb021	120,4706	31,8754	0,4549	0,0390	0,0577
mdb022	120,4706	32,5779	0,4449	0,0374	0,0568
mdb023	120,4706	31,7509	0,4599	0,0441	0,0524
mdb024	120,4706	31,3332	0,4455	0,0341	0,0521
mdb025	120,4706	31,8167	0,4557	0,0464	0,0558
mdb026	120,4706	32,0834	0,4403	0,0389	0,0553
mdb027	120,4706	31,5661	0,4246	0,0566	0,0658
mdb028	120,4706	31,9790	0,4369	0,0447	0,0523
mdb029	120,4706	32,4399	0,4495	0,0460	0,0534
mdb030	120,4706	34,2932	0,5168	0,0344	0,0578
average	120,4706	27,2473	0,4337	0,0401	0,0500

Table 7.13: Quadtree method average results,max block size=8,min block size=2

Threshold	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
th=2	4,6464	11,4130	47,5067	1,85E-05	0,9915
th=5	8,4638	19,5581	44,4661	3,69E-05	0,9784
th=10	17,0412	34,5981	41,4561	7,29E-05	0,9614
th=15	21,8802	42,7046	40,8295	8,42E-05	0,9595
th=20	23,9465	46,6388	40,5621	8,93E-05	0,9592
th=30	25,3122	49,4884	40,2256	9,64E-05	0,9584

Table 7.14: Quadtree method average results,max block size=16,min block size=2

Threshold	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
th=2	4,9857	12,6505	47,4635	1,86E-05	0,9914
th=5	9,6659	23,1781	44,2839	3,85E-05	0,9771
th=10	24,1591	49,9682	40,3978	9,34E-05	0,9541
th=15	39,1550	77,3337	38,1417	1,57E-04	0,9403
th=20	51,0120	102,2006	36,7615	2,16E-04	0,9303
th=30	64,2455	135,7385	35,3904	2,96E-04	0,9194

Table 7.15: Quadtree method average results,max block size=4,min block size=2

Threshold	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
th=2	13,2727	33,2195	34,1766	4,99E-04	0,9498
th=5	15,1889	36,6110	34,1553	5,01E-04	0,9488
th=10	21,3898	45,5308	34,1087	5,03E-04	0,9484
th=20	27,1322	55,4249	34,1954	4,98E-04	0,9505
th=30	28,0504	57,3847	34,2065	4,98-04	0,9503

Table 7.16: Quadtree method average time results,max block size=8,min block size=2

name	Compression Rate	Arithmetic Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
th=2	4,6464	11,4130	98,6914	31,5504	0,9976	1,1079
th=5	8,4638	19,5581	52,3414	19,3488	0,5425	0,5782
th=10	17,0412	34,5981	24,2881	11,2803	0,2397	0,2617
th=15	21,8802	42,7046	21,5363	10,3073	0,2048	0,2238
th=20	23,9465	46,6388	20,7606	10,0939	0,1858	0,2029
th=30	25,3122	49,4884	20,3485	10,0960	0,1853	0,1969

Table 7.17: Quadtree method average time results,max block size=16,min block size=2

Threshold	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
th=2	4,9857	97,7883	29,1511	0,8833	0,9447
th=5	9,6659	60,0336	19,5474	0,5053	0,5661
th=10	24,1591	38,7090	12,3693	0,3001	0,3162
th=15	39,1550	28,2064	9,4606	0,1527	0,1623
th=20	51,0120	24,9907	8,4495	0,1080	0,1146
th=30	64,2455	23,1812	7,6168	0,0812	0,0847

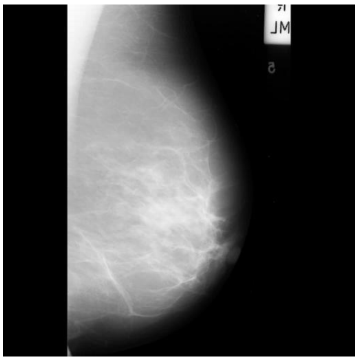


Figure 7.8: mdb014 Quadtree based reconstructed image,max block size=16, min block size=2, th=2,CR=4,588673, PSNR=47,258536, MSE=0,000019, ssim=0,991265

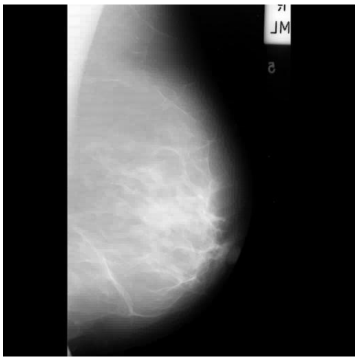


Figure 7.9: mdb014 Quadtree based reconstructed image,max block size=16, min block size=5, th=5, CR=10,917947, PSNR=43,801308, MSE=0,000042, ssim=0,975211

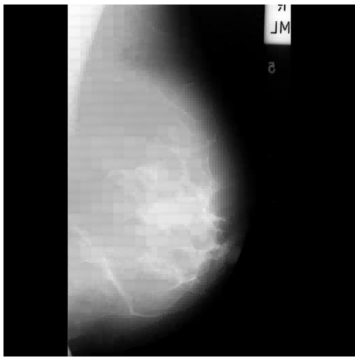


Figure 7.10: mdb014 Quadtree based reconstructed image,max block size=16, min block size=5, th=10, CR=27,334784, PSNR=40,315812, MSE=0,000093, ssim=0,956746

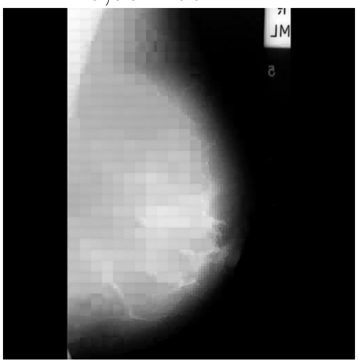


Figure 7.11: mdb014 Quadtree based reconstructed image,max block size=16, min block size=5, th=15, CR=41,313423, PSNR=38,145787, MSE=0,000153, ssim=0,945913

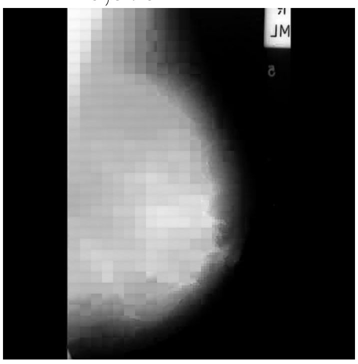


Figure 7.12: mdb014 Quadtree based reconstructed image,max block size=16, min block size=5, th=20, CR=48,875548, PSNR=36,863001, MSE=0,000206, ssim=0,937697

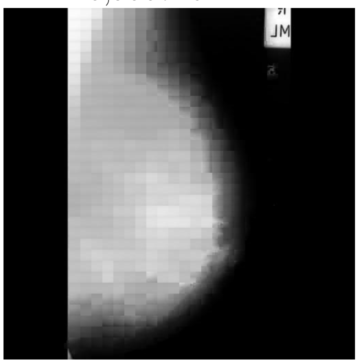


Figure 7.13: mdb014 Quadtree based reconstructed image, max block size=16, min block size=5, th=30, CR=57,645739, PSNR=35,860321, MSE=0,000259, ssim=0,930412

Figure 7.14: Threshold increasing effect on Quadtree Method

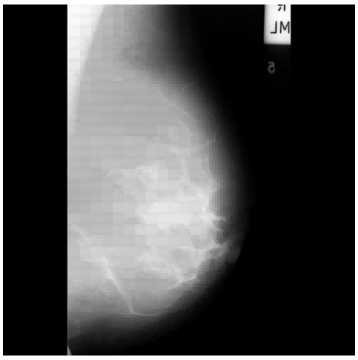


Figure 7.15: mdb014 Quadtree based reconstructed image,max block size=16,min block size=2, th=10, CR=40,315812, PSNR=43,801308, MSE=0,000093, ssim=0,956746

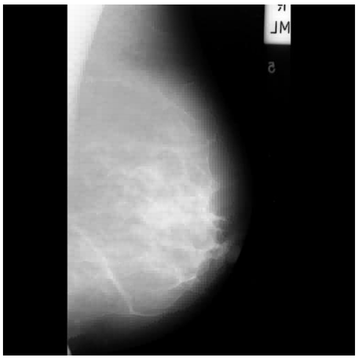


Figure 7.16: mdb014 Quadtree based reconstructed image, max block size=8, min block size=2, th=10, CR=18,503357, PSNR=41,885735, MSE=0,000065, ssim=0,965658

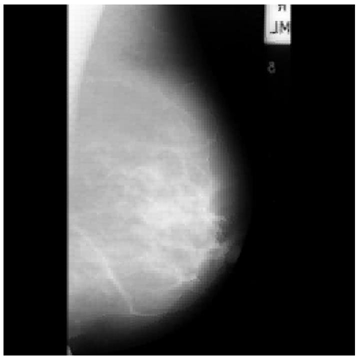


Figure 7.17: mdb014 Quadtree based reconstructed image, max block size=8, min block size=4, th=10, CR=27,100589, PSNR=31,111927, MSE=0,000774, ssim=0,952584

Figure 7.18: Block size effect on Quadtree base reconstruct image at th=10

Table 7.18: Quadtree method average time results, max block size = 4, min block size = 2

Threshold	Compression Rate	Arithmetic Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
th=2	13,2727	33,2195	41,2297	16,4401	0,3486	0,3761
th=5	15,1889	36,6110	37,2789	15,1836	0,3102	0,3537
th=10	21,3898	45,5308	29,9769	13,1416	0,3061	0,3362
th=20	27,1322	55,4249	26,4536	12,2013	0,2641	0,2808
th=30	28,0504	57,3847	25,9713	11,9575	0,2463	0,2663

Table 7.19: Quadtree method results, th=2, min block size=2, max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	4,8627	11,3750	46,5777	2,20E-05	0,9925
mdb012	4,2584	10,5019	45,4941	2,80E-05	0,9915
mdb013	4,7449	11,3094	47,9139	1,60E-05	0,9925
mdb014	4,3503	10,7873	47,2644	1,90E-05	0,9913
mdb015	5,4570	12,7603	48,6501	1,40E-05	0,9934
mdb016	4,9174	12,0779	47,4195	1,80E-05	0,9932
mdb017	7,0869	16,6532	50,1938	1,00E-05	0,9953
mdb018	7,0252	17,1977	49,3113	1,20E-05	0,9947
mdb019	3,4948	8,6884	46,2344	2,40E-05	0,9887
mdb020	3,6878	9,2864	45,8538	2,60E-05	0,9895
mdb021	4,3660	10,5992	46,8182	2,10E-05	0,9905
mdb022	5,0133	12,6754	47,7470	1,70E-05	0,9919
mdb023	4,0833	9,9987	47,6124	1,70E-05	0,9902
mdb024	4,5077	11,2203	48,1448	1,50E-05	0,9913
mdb025	3,6152	9,0692	47,4058	1,80E-05	0,9899
mdb026	3,9172	10,0421	47,5290	1,80E-05	0,9899
mdb027	3,8850	9,8289	48,3448	1,50E-05	0,9910
mdb028	3,8858	9,9834	48,0453	1,60E-05	0,9904
mdb029	4,7863	11,3987	46,0155	2,50E-05	0,9915
mdb030	4,9823	12,8069	47,5581	1,80E-05	0,9917
average	4,6464	11,4130	47,5067	1,85E-05	0,9915

Table 7.20: Quadtree method results, th=5, min block size=2, max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	9,2295	19,9854	43,9890	4,00E-05	0,9791
mdb012	8,3420	19,1088	43,0858	4,90E-05	0,9766
mdb013	9,3704	20,7046	44,5815	3,50E-05	0,9790
mdb014	9,5312	21,6946	43,9696	4,00E-05	0,9762
mdb015	8,2566	18,7861	45,8340	2,60E-05	0,9852
mdb016	7,8695	18,6666	44,8744	3,30E-05	0,9833
mdb017	10,4385	23,9003	47,3029	1,90E-05	0,9893
mdb018	10,2639	24,2482	46,9524	2,00E-05	0,9891
mdb019	6,3661	15,0491	43,1120	4,90E-05	0,9708
mdb020	6,9989	16,7106	43,1350	4,90E-05	0,9732
mdb021	7,2024	16,7237	44,1199	3,90E-05	0,9781
mdb022	7,9532	19,4377	45,1267	3,10E-05	0,9824
mdb023	8,5448	19,1722	44,0664	3,90E-05	0,9746
mdb024	9,0687	20,8715	44,6075	3,50E-05	0,9778
mdb025	6,7187	15,8523	43,7426	4,20E-05	0,9724
mdb026	7,3060	17,6230	43,9825	4,00E-05	0,9740
mdb027	8,8254	20,2941	44,2290	3,80E-05	0,9739
mdb028	8,6020	20,1942	44,1923	3,80E-05	0,9739
mdb029	9,1454	19,9249	43,7544	4,20E-05	0,9794
mdb030	9,2420	22,2147	44,6648	3,40E-05	0,9803
average	8,4638	19,5581	44,4661	3,69E-05	0,9784

Table 7.21: Quadtree method results, th=10, min block size=2, max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	17,2173	32,5518	41,6068	6,90E-05	0,9652
mdb012	15,8869	31,8828	41,1667	7,60E-05	0,9640
mdb013	17,9933	34,4609	42,1038	6,20E-05	0,9667
mdb014	18,5034	36,3590	41,8857	6,50E-05	0,9657
mdb015	15,9919	32,4265	41,7334	6,70E-05	0,9653
mdb016	16,2511	34,6551	41,1914	7,60E-05	0,9631
mdb017	17,4659	37,1710	43,4667	4,50E-05	0,9773
mdb018	17,6307	38,9856	43,4040	4,60E-05	0,9779
mdb019	14,7633	29,5991	39,8967	1,02E-04	0,9447
mdb020	15,8441	32,1127	40,2284	9,50E-05	0,9511
mdb021	15,1774	30,5707	40,6390	8,60E-05	0,9554
mdb022	17,4289	38,0318	41,2468	7,50E-05	0,9611
mdb023	17,1384	33,2976	41,2924	7,40E-05	0,9587
mdb024	18,6460	37,0155	41,8987	6,50E-05	0,9636
mdb025	16,0732	32,1102	40,3097	9,30E-05	0,9470
mdb026	16,9756	35,2303	40,6388	8,60E-05	0,9511
mdb027	18,9031	37,1954	41,6590	6,80E-05	0,9597
mdb028	18,1601	36,7322	41,4931	7,10E-05	0,9587
mdb029	16,2190	31,4519	41,4782	7,10E-05	0,9669
mdb030	18,5534	40,1223	41,7821	6,60E-05	0,9659
average	17,0412	34,5981	41,4561	7,29E-05	0,9614

Table 7.22: Quadtree method results, th=15, min block size=2, max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	20,5569	37,5155	41,2955	0,0001	9,65E-01
mdb012	18,7651	36,3647	41,0193	0,0001	9,65E-01
mdb013	22,1874	41,1529	41,4770	0,0001	9,65E-01
mdb014	22,4417	42,4344	41,5752	0,0001	9,66E-01
mdb015	21,5819	41,5146	40,8625	0,0001	9,62E-01
mdb016	23,8750	48,6228	40,2143	0,0001	9,59E-01
mdb017	21,5705	44,9079	42,3075	0,0001	9,74E-01
mdb018	22,2715	49,0459	42,0199	0,0001	9,76E-01
mdb019	20,1341	38,0339	39,3244	0,0001	9,42E-01
mdb020	20,8699	39,6925	39,8110	0,0001	9,50E-01
mdb021	20,4852	38,7687	39,9251	0,0001	9,52E-01
mdb022	24,0565	49,7616	40,5345	0,0001	9,59E-01
mdb023	21,2583	39,6235	41,0911	0,0001	9,59E-01
mdb024	22,3320	42,8725	41,8279	0,0001	9,65E-01
mdb025	22,1158	41,7543	39,5050	0,0001	9,42E-01
mdb026	23,8196	46,7874	39,8429	0,0001	9,46E-01
mdb027	23,6280	44,8406	41,1654	0,0001	9,58E-01
mdb028	23,2407	45,2323	40,7115	0,0001	9,55E-01
mdb029	19,4210	36,5867	40,7736	0,0001	9,65E-01
mdb030	22,9938	48,5800	41,3063	0,0001	9,65E-01
average	21,8802	42,7046	40,8295	0,0001	9,59E-01

Table 7.23: Quadtree method results, th=20, min block size=2, max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	21,6614	39,3056	41,2399	7,50E-05	0,9650
mdb012	19,6155	37,8404	40,9707	8,00E-05	0,9651
mdb013	23,7234	43,6688	41,4199	7,20E-05	0,9646
mdb014	23,6416	44,4670	41,4900	7,10E-05	0,9663
mdb015	23,9445	45,5507	40,5197	8,90E-05	0,9613
mdb016	26,8003	54,4786	40,1433	9,70E-05	0,9594
mdb017	23,3198	48,3906	41,8714	6,50E-05	0,9737
mdb018	25,5196	56,8858	40,9372	8,10E-05	0,9749
mdb019	22,2715	41,7169	39,1174	1,23E-04	0,9421
mdb020	22,6893	42,6103	39,7321	1,06E-04	0,9503
mdb021	22,5524	42,2966	39,6103	1,09E-04	0,9517
mdb022	28,4113	58,8460	40,1172	9,70E-05	0,9586
mdb023	22,6893	42,2192	40,9555	8,00E-05	0,9593
mdb024	23,4261	44,8934	41,7534	6,70E-05	0,9648
mdb025	24,2409	45,4746	39,4118	1,15E-04	0,9413
mdb026	25,6310	50,0072	39,7933	1,05E-04	0,9462
mdb027	24,8413	47,0741	41,0891	7,80E-05	0,9576
mdb028	25,6470	49,6838	40,2926	9,30E-05	0,9534
mdb029	21,1490	39,4639	40,4140	9,10E-05	0,9639
mdb030	27,1543	57,9020	40,3628	9,20E-05	0,9644
average	23,9465	46,6388	40,5621	8,93E-05	0,9592

Table 7.24: Quadtree method results, th=30, min block size=2, max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	22,3563	40,6000	40,8860	8,20E-05	0,9643
mdb012	20,0850	38,6543	40,7591	8,40E-05	0,9650
mdb013	24,5447	45,2265	41,2739	7,50E-05	0,9644
mdb014	24,7516	46,6698	41,1212	7,70E-05	0,9660
mdb015	26,7829	51,0828	40,3200	9,30E-05	0,9608
mdb016	28,0815	57,0545	40,0929	9,80E-05	0,9594
mdb017	25,7594	53,8491	41,2627	7,50E-05	0,9727
mdb018	27,5916	61,9232	40,5788	8,80E-05	0,9744
mdb019	23,5199	44,0569	38,9030	1,29E-04	0,9415
mdb020	23,7783	44,6469	39,4976	1,12E-04	0,9497
mdb021	23,6009	44,1850	39,5147	1,12E-04	0,9517
mdb022	29,9204	62,7008	40,0502	9,90E-05	0,9584
mdb023	25,1451	47,3334	39,1448	1,22E-04	0,9540
mdb024	24,9015	47,8693	40,8676	8,20E-05	0,9631
mdb025	25,1451	47,2140	39,2610	1,19E-04	0,9406
mdb026	26,3369	51,3555	39,7532	1,06E-04	0,9461
mdb027	25,3779	48,1960	40,7099	8,50E-05	0,9568
mdb028	26,7654	51,9856	39,8258	1,04E-04	0,9521
mdb029	22,3078	41,4048	40,1784	9,60E-05	0,9636
mdb030	29,4912	63,7607	40,5122	8,90E-05	0,9642
average	25,3122	49,4884	40,2256	9,64E-05	0,9584

Table 7.25: Quadtree method results, th=2, min block size=2, max block size=16

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb017	8,1357	19,9215	50,1375	1,00E-05	0,9949
mdb018	8,0544	20,6275	49,2589	1,20E-05	0,9943
mdb015	5,9722	14,5636	48,6243	1,40E-05	0,9933
mdb027	4,0329	10,4340	48,3058	1,50E-05	0,9909
mdb024	4,7569	12,2200	48,0937	1,60E-05	0,9912
mdb028	4,0361	10,6161	48,0219	1,60E-05	0,9903
mdb013	5,0739	12,4976	47,8993	1,60E-05	0,9923
mdb022	5,4275	14,3628	47,7194	1,70E-05	0,9918
mdb023	4,2650	10,7057	47,5887	1,70E-05	0,9901
mdb030	5,3729	14,4528	47,5236	1,80E-05	0,9916
mdb026	4,0882	10,7580	47,5141	1,80E-05	0,9897
mdb016	5,3016	13,4812	47,4061	1,80E-05	0,9931
mdb025	3,7419	9,5401	47,3831	1,80E-05	0,9897
mdb014	4,5887	11,7535	47,2585	1,90E-05	0,9913
mdb021	4,6129	11,5140	46,7464	2,10E-05	0,9898
mdb011	5,2175	12,6216	46,5336	2,20E-05	0,9919
mdb019	3,6193	9,1382	46,2275	2,40E-05	0,9886
mdb029	5,0978	12,5684	45,9591	2,50E-05	0,9913
mdb020	3,8379	9,8969	45,8335	2,60E-05	0,9894
mdb012	4,4802	11,3368	45,2343	3,00E-05	0,9908
average	4,9857	12,6505	47,4635	1,86E-05	0,9913

Table 7.26: Quadtree method results, th=5, min block size=2, max block size=16

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb017	12,9459	30,7911	47,1507	1,90E-05	0,9887
mdb018	12,6824	31,1951	46,8726	2,10E-05	0,9886
mdb015	9,5312	22,8023	45,7545	2,70E-05	0,9848
mdb022	9,1291	23,7393	45,0429	3,10E-05	0,9825
mdb016	8,9679	22,1921	44,8050	3,30E-05	0,9834
mdb030	10,8231	27,4791	44,5296	3,50E-05	0,9804
mdb024	10,3622	25,0409	44,4707	3,60E-05	0,9768
mdb013	10,8345	24,7510	44,4611	3,60E-05	0,9775
mdb028	9,5444	23,2024	43,9323	4,00E-05	0,9726
mdb011	10,6881	23,8970	43,8675	4,10E-05	0,9778
mdb021	7,9701	19,1196	43,8384	4,10E-05	0,9759
mdb026	8,0215	20,0188	43,8042	4,20E-05	0,9730
mdb014	10,9179	25,8885	43,8013	4,20E-05	0,9752
mdb027	9,8156	23,2027	43,7828	4,20E-05	0,9694
mdb023	9,5800	22,0751	43,7380	4,20E-05	0,9710
mdb025	7,2429	17,4681	43,4907	4,50E-05	0,9702
mdb029	10,5374	23,7087	43,4495	4,50E-05	0,9757
mdb020	7,6032	18,7251	43,0638	4,90E-05	0,9725
mdb019	6,8303	16,3992	42,9885	5,00E-05	0,9699
mdb012	9,2900	21,8665	42,8329	5,20E-05	0,9755
average	9,6659	23,1781	44,2839	3,85E-05	0,9771

Table 7.27: Quadtree method results,th=10, min block size=2,max block size=16

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb017	26,3706	56,0766	42,8824	5,10E-05	0,9745
mdb018	26,8879	59,7071	42,7002	5,40E-05	0,9752
mdb015	22,1277	46,4105	41,2483	7,50E-05	0,9620
mdb022	25,0379	57,1073	40,8947	8,10E-05	0,9591
mdb016	22,6145	49,6732	40,7903	8,30E-05	0,9606
mdb024	27,6101	56,3811	40,7256	8,50E-05	0,9573
mdb013	26,6096	51,2150	40,6130	8,70E-05	0,9576
mdb030	28,1199	62,3429	40,6013	8,70E-05	0,9602
mdb011	25,1759	48,0954	40,4768	9,00E-05	0,9573
mdb014	27,3348	55,0188	40,3158	9,30E-05	0,9567
mdb027	27,3894	54,5565	39,9956	1,00E-04	0,9455
mdb012	21,3910	43,8781	39,9858	1,00E-04	0,9563
mdb028	26,1194	54,6489	39,8831	1,03E-04	0,9480
mdb029	22,9810	44,8723	39,8514	1,03E-04	0,9555
mdb023	24,4572	47,8758	39,8344	1,04E-04	0,9476
mdb021	19,9292	40,8086	39,8330	1,04E-04	0,9484
mdb026	22,9810	48,8710	39,7675	1,05E-04	0,9444
mdb025	20,6602	41,5253	39,4377	1,14E-04	0,9379
mdb020	20,8065	43,2679	39,1791	1,21E-04	0,9421
mdb019	18,5786	37,0312	38,9403	1,28E-04	0,9357
average	24,1591	49,9682	40,3978	9,34E-05	0,9541

Table 7.28: Quadtree method results,th=15,min block size=2, max block size=16

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb017	39,3750	80,6318	40,1911	9,60E-05	0,9634
mdb018	42,2039	93,5936	39,8773	1,03E-04	0,9661
mdb011	34,9864	63,6310	39,1111	1,23E-04	0,9495
mdb022	45,8254	97,1084	38,5905	1,38E-04	0,9469
mdb024	40,7404	80,3753	38,5694	1,39E-04	0,9470
mdb013	40,6197	74,8315	38,5524	1,40E-04	0,9457
mdb015	38,3111	75,5349	38,4414	1,43E-04	0,9469
mdb012	28,8498	57,3557	38,3860	1,45E-04	0,9477
mdb016	45,0719	91,9925	38,3561	1,46E-04	0,9462
mdb030	44,2474	97,2164	38,3356	1,47E-04	0,9500
mdb014	41,3134	79,7214	38,1458	1,53E-04	0,9459
mdb023	36,8685	68,2978	37,9866	1,59E-04	0,9375
mdb021	32,8090	61,5506	37,8621	1,64E-04	0,9347
mdb027	45,6727	85,8047	37,8065	1,66E-04	0,9285
mdb029	32,1422	60,8682	37,7509	1,68E-04	0,9436
mdb028	44,2951	89,4499	37,6358	1,72E-04	0,9304
mdb026	44,8263	89,9602	37,1288	1,94E-04	0,9235
mdb020	34,7792	68,0385	36,9258	2,03E-04	0,9249
mdb025	38,4904	72,2932	36,6649	2,16E-04	0,9122
mdb019	31,6719	58,4181	36,5161	2,23E-04	0,9150
average	39,1550	77,3337	38,1417	1,57E-04	0,9403

Table 7.29: Quadtree method results,th=20,min block size=2,max block size=16

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb017	47,5750	97,2840	38,8924	1,29E-04	0,9575
mdb018	58,0527	132,5885	38,3977	1,45E-04	0,9607
mdb011	39,7942	71,9089	38,3164	1,47E-04	0,9446
mdb013	48,7596	88,6445	37,6656	1,71E-04	0,9400
mdb012	32,4979	65,0885	37,2246	1,89E-04	0,9398
mdb029	38,1689	70,8306	37,1886	1,91E-04	0,9394
mdb024	48,0194	96,7812	37,0342	1,98E-04	0,9371
mdb023	43,5908	80,8151	36,9337	2,03E-04	0,9301
mdb015	49,4635	96,4208	36,9037	2,04E-04	0,9373
mdb030	67,5956	152,3539	36,8795	2,05E-04	0,9429
mdb014	48,8755	94,6197	36,8630	2,06E-04	0,9377
mdb022	76,2434	167,3705	36,4027	2,29E-04	0,9329
mdb027	56,7703	107,9615	36,3060	2,34E-04	0,9158
mdb016	65,9689	137,4100	36,1622	2,42E-04	0,9307
mdb028	61,4352	127,8751	36,0445	2,49E-04	0,9166
mdb021	42,1174	78,8255	36,0306	2,49E-04	0,9198
mdb020	42,9982	83,2170	35,9346	2,55E-04	0,9168
mdb026	60,0886	123,3546	35,5606	2,78E-04	0,9078
mdb025	51,4437	96,1379	35,3957	2,89E-04	0,8978
mdb019	40,7808	74,5230	35,0939	3,09E-04	0,9006
average	51,0120	102,2006	36,7615	2,16E-04	0,9303

Table 7.30: Quadtree method results,th=30,min block size=2,max block size=16

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb018	75,4046	178,6330	37,0936	1,95E-04	0,9535
mdb017	63,0362	131,4582	37,0822	1,96E-04	0,9485
mdb011	45,4204	83,9532	36,6083	2,18E-04	0,9338
mdb013	57,0064	105,3422	36,5214	2,23E-04	0,9315
mdb029	44,2474	81,7794	36,1672	2,42E-04	0,9325
mdb014	57,6457	113,3106	35,8603	2,59E-04	0,9304
mdb012	35,7157	72,9241	35,7437	2,66E-04	0,9296
mdb015	71,4727	141,7953	35,6556	2,72E-04	0,9267
mdb024	59,9135	125,8568	35,5624	2,78E-04	0,9257
mdb030	99,9691	257,9523	35,5552	2,78E-04	0,9357
mdb027	64,1174	125,3378	35,1560	3,05E-04	0,9062
mdb016	83,1840	177,2591	35,0567	3,12E-04	0,9212
mdb023	59,8263	115,0385	34,9781	3,18E-04	0,9135
mdb022	107,8338	275,6509	34,7716	3,33E-04	0,9200
mdb021	50,4961	96,2394	34,6798	3,40E-04	0,9063
mdb028	74,9920	166,1111	34,6230	3,45E-04	0,9044
mdb020	52,3607	104,0048	34,4685	3,57E-04	0,9024
mdb026	70,7350	151,4408	34,3636	3,66E-04	0,8961
mdb025	61,5271	117,9169	34,1547	3,84E-04	0,8844
mdb019	50,0048	92,7656	33,7060	4,26E-04	0,8850
average	64,2455	135,7385	35,3904	2,96E-04	0,9194

Table 7.31: Quadtree method results,th=2,min block size=4,max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	13,8252	32,6624	32,3333	5,84E-04	0,9532
mdb012	12,5931	32,2421	28,5955	1,38E-03	0,9367
mdb013	13,5384	32,3156	33,4391	4,53E-04	0,9560
mdb014	12,5126	30,9904	31,0912	7,78E-04	0,9500
mdb015	15,2732	35,1724	37,8745	1,63E-04	0,9606
mdb016	14,3461	34,8996	36,9310	2,03E-04	0,9574
mdb017	17,6307	42,0161	40,5298	8,90E-05	0,9735
mdb018	17,6231	42,8200	40,6168	8,70E-05	0,9745
mdb019	11,1974	29,7295	31,8835	6,48E-04	0,9300
mdb020	11,6610	30,3517	31,2900	7,43E-04	0,9359
mdb021	12,7730	31,8319	31,9630	6,36E-04	0,9406
mdb022	14,5491	36,0763	39,5681	1,10E-04	0,9572
mdb023	11,9839	30,4650	32,8314	5,21E-04	0,9468
mdb024	12,8009	32,7307	32,1377	6,11E-04	0,9490
mdb025	11,1791	29,9119	33,0104	5,00E-04	0,9321
mdb026	11,7880	31,6766	31,4642	7,14E-04	0,9334
mdb027	11,2341	29,5603	33,8352	4,14E-04	0,9473
mdb028	11,2896	31,0989	32,4620	5,67E-04	0,9412
mdb029	13,4675	32,1452	31,7069	6,75E-04	0,9559
mdb030	14,1877	35,6932	39,9685	1,01E-04	0,9654
average	13,2727	33,2195	34,1766	4,99E-04	0,9498

Table 7.32: Quadtree method results,th=5,min block size=4,max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	16,0419	36,3118	32,3166	5,87E-04	0,9514
mdb012	14,2714	34,9176	28,5906	1,38E-03	0,9358
mdb013	15,5917	35,8169	33,4228	4,55E-04	0,9547
mdb014	15,2846	36,2810	31,0778	7,80E-04	0,9485
mdb015	15,9795	36,7046	37,8523	1,64E-04	0,9601
mdb016	15,4164	36,9152	36,9114	2,04E-04	0,9567
mdb017	18,5870	43,9019	40,4961	8,90E-05	0,9731
mdb018	18,5618	44,7135	40,5814	8,70E-05	0,9741
mdb019	12,4936	32,2312	31,8695	6,50E-04	0,9285
mdb020	13,1741	33,1933	31,2794	7,45E-04	0,9346
mdb021	14,3862	34,6093	31,9507	6,38E-04	0,9395
mdb022	15,5387	38,2141	39,5339	1,11E-04	0,9565
mdb023	15,0386	35,8273	32,8051	5,24E-04	0,9449
mdb024	15,4337	37,1381	32,1259	6,13E-04	0,9481
mdb025	12,8932	32,6415	32,9992	5,01E-04	0,9314
mdb026	13,8345	35,4195	31,4566	7,15E-04	0,9328
mdb027	14,6008	35,0243	33,8188	4,15E-04	0,9465
mdb028	14,8218	36,8083	32,4498	5,69E-04	0,9406
mdb029	15,9609	36,5631	31,6868	6,78E-04	0,9540
mdb030	15,8685	38,9878	39,8809	1,03E-04	0,9641
average	15,1889	36,6110	34,1553	5,01E-04	0,9488

Table 7.33: Quadtree method results,th=10,min block size=4,max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	21,9271	44,2643	32,3001	5,89E-04	0,9505
mdb012	20,9230	44,9570	28,5871	1,39E-03	0,9360
mdb013	22,2474	44,7603	33,4081	4,56E-04	0,9547
mdb014	22,6269	46,7072	31,0697	7,82E-04	0,9491
mdb015	20,6706	43,0521	37,7463	1,68E-04	0,9578
mdb016	20,3534	44,5132	36,8576	2,06E-04	0,9557
mdb017	22,5031	50,0346	40,3882	9,10E-05	0,9720
mdb018	22,4785	51,0703	40,4567	9,00E-05	0,9732
mdb019	19,1051	41,4023	31,8448	6,54E-04	0,9277
mdb020	20,1046	43,1371	31,2522	7,50E-04	0,9331
mdb021	19,7001	41,7909	31,9334	6,41E-04	0,9391
mdb022	20,6291	45,7454	39,4123	1,14E-04	0,9556
mdb023	21,2363	44,3889	32,7843	5,27E-04	0,9446
mdb024	22,4662	47,4737	32,1113	6,15E-04	0,9483
mdb025	20,4445	42,8953	32,9891	5,02E-04	0,9320
mdb026	20,8065	45,2314	31,4480	7,16E-04	0,9332
mdb027	22,6893	46,5196	33,8238	4,15E-04	0,9483
mdb028	22,4662	47,6810	32,4476	5,69E-04	0,9417
mdb029	21,7416	45,0013	31,6582	6,83E-04	0,9524
mdb030	22,6768	49,9893	39,6563	1,08E-04	0,9621
average	21,3898	45,5308	34,1087	5,03E-04	0,9484

Table 7.34: Quadtree method results,th=15,min block size=4,max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	25,2998	49,4658	32,3243	5,86E-04	0,9523
mdb012	24,3126	50,5167	28,6024	1,38E-03	0,9386
mdb013	25,7433	50,2962	33,4160	4,55E-04	0,9555
mdb014	26,0532	52,1537	31,0970	7,77E-04	0,9518
mdb015	25,1913	49,9167	37,7812	1,67E-04	0,9586
mdb016	25,7111	53,3165	36,9196	2,03E-04	0,9570
mdb017	25,6470	56,0466	40,3512	9,20E-05	0,9718
mdb018	25,5513	57,2554	40,5053	8,90E-05	0,9739
mdb019	24,3126	49,5055	31,8694	6,50E-04	0,9302
mdb020	25,0991	50,9129	31,2783	7,45E-04	0,9356
mdb021	24,7964	49,5605	31,9529	6,38E-04	0,9411
mdb022	25,8729	54,1061	39,5012	1,12E-04	0,9574
mdb023	25,5513	51,3555	32,8185	5,23E-04	0,9472
mdb024	26,3706	54,1872	32,1434	6,10E-04	0,9511
mdb025	25,4564	50,7760	32,9987	5,01E-04	0,9328
mdb026	26,3200	54,2026	31,4517	7,16E-04	0,9340
mdb027	26,8353	53,7828	33,8436	4,13E-04	0,9495
mdb028	26,2193	53,7773	32,4599	5,68E-04	0,9428
mdb029	24,3126	49,2451	31,6706	6,81E-04	0,9536
mdb030	25,9545	55,9464	39,7779	1,05E-04	0,9638
average	25,5305	52,3163	34,1382	5,01E-04	0,9499

Table 7.35: Quadtree method results,th=20,min block size=4,max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	26,4725	51,6591	32,3399	5,83E-04	0,9530
mdb012	25,3466	52,4091	28,6100	1,38E-03	0,9393
mdb013	27,3166	53,1894	33,4368	4,53E-04	0,9560
mdb014	27,1006	54,1508	31,1119	7,74E-04	0,9526
mdb015	26,7829	52,6420	37,8226	1,65E-04	0,9592
mdb016	28,1007	58,0768	36,9709	2,01E-04	0,9577
mdb017	26,5752	57,9676	40,4086	9,10E-05	0,9722
mdb018	26,7480	59,9255	40,5914	8,70E-05	0,9744
mdb019	26,3200	53,2016	31,8882	6,47E-04	0,9312
mdb020	26,8003	53,9516	31,2984	7,42E-04	0,9367
mdb021	26,4896	52,6962	31,9681	6,36E-04	0,9419
mdb022	28,8904	60,0043	39,8416	1,04E-04	0,9583
mdb023	26,8353	53,8270	32,8273	5,22E-04	0,9479
mdb024	27,3894	56,3024	32,1558	6,09E-04	0,9517
mdb025	27,3894	54,6020	33,0177	4,99E-04	0,9332
mdb026	28,0623	57,7473	31,4586	7,15E-04	0,9343
mdb027	28,0241	56,4206	33,8590	4,11E-04	0,9500
mdb028	28,0432	57,3071	32,4494	5,69E-04	0,9426
mdb029	25,9709	52,4314	31,6718	6,80E-04	0,9536
mdb030	27,9859	59,9872	40,1800	9,60E-05	0,9641
average	27,1322	55,4249	34,1954	4,98E-04	0,9505

Table 7.36: Quadtree method results,th=30,min block size=4,max block size=8

name	Compression Rate	Arithmetic Compression Rate	psnr	mse	ssim
mdb011	26,9232	52,5049	32,3292	5,85E-04	0,9527
mdb012	25,7756	53,2569	28,6172	1,38E-03	0,9395
mdb013	27,9479	54,4687	33,4657	4,50E-04	0,9563
mdb014	27,9669	55,9196	31,1161	7,73E-04	0,9528
mdb015	28,6488	56,2028	37,8240	1,65E-04	0,9592
mdb016	29,3230	60,7043	36,9546	2,02E-04	0,9577
mdb017	27,6844	60,2353	40,3632	9,20E-05	0,9719
mdb018	27,7218	62,4115	40,5370	8,80E-05	0,9744
mdb019	27,4259	55,3601	31,9016	6,45E-04	0,9313
mdb020	27,4993	55,2959	31,2984	7,42E-04	0,9366
mdb021	27,4259	54,5877	31,9792	6,34E-04	0,9422
mdb022	29,9859	62,8605	39,9921	1,00E-04	0,9584
mdb023	28,1007	56,3902	32,7171	5,35E-04	0,9459
mdb024	28,0623	57,7092	32,1190	6,14E-04	0,9509
mdb025	28,2357	56,5377	33,0109	5,00E-04	0,9329
mdb026	28,6888	59,1814	31,4606	7,14E-04	0,9343
mdb027	28,3330	57,0343	33,8430	4,13E-04	0,9498
mdb028	28,9107	59,1764	32,4097	5,74E-04	0,9419
mdb029	26,8353	54,0044	31,6830	6,79E-04	0,9539
mdb030	29,5124	63,8519	40,5086	8,90E-05	0,9642
average	28,0504	57,3847	34,2065	4,98E-04	0,9503

Table 7.37: Quadtree time results, th=2, min block size=2, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	4,8627	95,7455	29,9025	1,0459	1,1344
mdb012	4,2584	105,9455	32,8547	1,0058	1,1392
mdb013	4,7449	93,2147	30,6712	1,0331	1,1414
mdb014	4,3503	103,6046	32,0469	1,0265	1,1627
mdb015	5,4570	81,7437	27,4791	0,9534	1,0744
mdb016	4,9174	92,6730	30,5157	0,9929	1,1471
mdb017	7,0869	68,9790	22,3579	0,7063	0,7834
mdb018	7,0252	67,6180	22,6835	0,6581	0,7329
mdb019	3,4948	125,1190	36,6729	1,2157	1,3676
mdb020	3,6878	117,2035	36,8183	1,1690	1,2853
mdb021	4,3660	98,2089	31,1894	1,2891	1,1655
mdb022	5,0133	88,3170	30,5707	0,8248	0,9254
mdb023	4,0833	104,2306	34,5068	1,0633	1,1769
mdb024	4,5077	98,3257	32,9103	0,9572	1,1337
mdb025	3,6152	117,0552	36,7856	1,1634	1,2900
mdb026	3,9172	113,8944	34,9584	1,0536	1,2040
mdb027	3,8850	108,6941	34,9605	1,0499	1,2460
mdb028	3,8858	112,1945	33,7969	0,8845	1,0077
mdb029	4,7863	90,3038	30,3609	1,1010	1,1950
mdb030	4,9823	90,7587	28,9648	0,7584	0,8458
average	4,6464	98,6914	31,5504	0,9976	1,1079

Table 7.38: Quadtree time results, th=5, min block size=2, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetik decoding time
mdb011	9,2295	52,5165	19,4709	0,6443	0,7024
mdb012	8,3420	57,4048	21,4363	0,6898	0,7369
mdb013	9,3704	55,7435	19,7829	0,5858	0,6382
mdb014	9,5312	50,9023	19,0593	0,6143	0,6267
mdb015	8,2566	58,0015	21,9847	0,7480	0,7182
mdb016	7,8695	62,8306	21,3242	0,7635	0,6857
mdb017	10,4385	48,6525	17,6307	0,5108	0,6328
mdb018	10,2639	50,2362	18,2654	0,4688	0,5242
mdb019	6,3661	72,3128	24,8856	0,7582	0,8480
mdb020	6,9989	66,0480	22,9854	0,6778	0,7577
mdb021	7,2024	67,9675	23,1868	0,8404	0,8157
mdb022	7,9532	64,0757	21,7224	0,6320	0,7137
mdb023	8,5448	60,5329	18,1245	0,3738	0,4049
mdb024	9,0687	39,0600	16,7036	0,3614	0,3851
mdb025	6,7187	46,0534	18,1681	0,4569	0,4706
mdb026	7,3060	42,5435	17,6382	0,3636	0,4100
mdb027	8,8254	39,0529	17,2851	0,3410	0,3756
mdb028	8,6020	38,3414	15,3270	0,3362	0,3674
mdb029	9,1454	35,6347	15,4558	0,3723	0,4068
mdb030	9,2420	38,9182	16,5402	0,3105	0,3440

Table 7.39: Quadtree time results, th=10, min block size=2, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	17,2173	24,1727	11,7973	0,2889	0,3328
mdb012	15,8869	28,1567	11,8563	0,2508	0,2794
mdb013	17,9933	24,3931	12,0214	0,2596	0,2700
mdb014	18,5034	24,3526	10,6106	0,2186	0,2463
mdb015	15,9919	24,7344	13,0609	0,2981	0,3069
mdb016	16,2511	26,2984	12,2291	0,2308	0,2503
mdb017	17,4659	24,5211	11,2759	0,2152	0,2379
mdb018	17,6307	24,9374	11,9401	0,2355	0,2526
mdb019	14,7633	28,8454	12,3323	0,2787	0,3117
mdb020	15,8441	25,8901	10,9923	0,2409	0,2653
mdb021	15,1774	24,4335	11,1119	0,2663	0,2793
mdb022	17,4289	24,2069	10,6388	0,1954	0,2086
mdb023	17,1384	22,8168	10,6556	0,2411	0,2631
mdb024	18,6460	21,7528	10,2987	0,2368	0,2637
mdb025	16,0732	23,7995	11,0280	0,2424	0,2707
mdb026	16,9756	22,8967	10,6992	0,2299	0,2391
mdb027	18,9031	21,8953	11,6060	0,2144	0,2378
mdb028	18,1601	22,2095	10,3399	0,2120	0,2367
mdb029	16,2190	23,4800	10,8721	0,2568	0,2818
mdb030	18,5534	21,9693	10,2400	0,1813	0,1993
average	17,0412	24,2881	11,2803	0,2397	0,2617

Table 7.40: Quadtree time results, th=15, min block size=2, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	20,5569	20,9666	9,8882	0,2259	0,2453
mdb012	18,7651	23,2382	10,3174	0,2251	0,2457
mdb013	22,1874	20,4913	10,6734	0,2045	0,2239
mdb014	22,4417	20,1592	10,1623	0,1935	0,2221
mdb015	21,5819	21,6263	9,7820	0,2034	0,2175
mdb016	23,8750	19,3747	9,4343	0,1598	0,1771
mdb017	21,5705	20,4406	9,8165	0,1742	0,1901
mdb018	22,2715	20,0584	9,6544	0,1498	0,1604
mdb019	20,1341	23,3432	10,8289	0,2673	0,3060
mdb020	20,8699	23,8075	10,8592	0,2204	0,2426
mdb021	20,4852	23,9859	12,1852	0,2669	0,2950
mdb022	24,0565	21,0726	10,1646	0,1617	0,1842
mdb023	21,2583	22,1370	10,1925	0,2248	0,2402
mdb024	22,3320	22,4784	10,8595	0,1958	0,2068
mdb025	22,1158	20,5934	10,2766	0,2058	0,2219
mdb026	23,8196	20,0779	9,7547	0,2217	0,2270
mdb027	23,6280	20,9324	9,8049	0,1866	0,2101
mdb028	23,2407	21,4079	10,0287	0,1856	0,2079
mdb029	19,4210	22,9248	10,8399	0,2293	0,2507
mdb030	22,9938	21,6089	10,6234	0,1938	0,2020
average	21,8802	21,5363	10,3073	0,2048	0,2238

Table 7.41: Quadtree time results, th=20, min block size=2, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	21,6614	23,2561	10,2794	0,2223	0,2415
mdb012	19,6155	23,4300	11,0084	0,2212	0,2429
mdb013	23,7234	20,3128	9,5306	0,2000	0,2167
mdb014	23,6416	19,3704	9,5256	0,2016	0,2066
mdb015	23,9445	19,2333	9,5051	0,1861	0,2053
mdb016	26,8003	19,2784	9,7309	0,1750	0,2060
mdb017	23,3198	21,5657	9,8857	0,1645	0,1797
mdb018	25,5196	21,1015	10,2080	0,1152	0,1300
mdb019	22,2715	21,7795	9,9451	0,2000	0,2189
mdb020	22,6893	21,3095	11,6438	0,2032	0,2384
mdb021	22,5524	21,5830	11,2738	0,2121	0,2234
mdb022	28,4113	20,6097	10,5639	0,1228	0,1340
mdb023	22,6893	21,4602	9,9908	0,2080	0,2275
mdb024	23,4261	20,4327	9,9673	0,1897	0,2091
mdb025	24,2409	20,0383	9,8976	0,1961	0,2187
mdb026	25,6310	19,8371	9,6780	0,1753	0,1858
mdb027	24,8413	19,9638	9,8086	0,2039	0,2038
mdb028	25,6470	19,7513	9,6420	0,1688	0,1900
mdb029	21,1490	21,6380	10,3163	0,2285	0,2481
mdb030	27,1543	19,2600	9,4764	0,1221	0,1311
average	23,9465	20,7606	10,0939	0,1858	0,2029

Table 7.42: Quadtree time results, th=30, min block size=2, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	22,3563	21,2479	10,2333	0,2196	0,2405
mdb012	20,0850	23,6667	10,7718	0,2214	0,2437
mdb013	24,5447	20,7505	10,0334	0,2030	0,2219
mdb014	24,7516	20,4421	10,4062	0,1850	0,1992
mdb015	26,7829	19,0802	9,5954	0,1713	0,1807
mdb016	28,0815	18,3446	9,2322	0,1686	0,1541
mdb017	25,7594	18,8910	10,9614	0,1854	0,1755
mdb018	27,5916	18,9845	9,4464	0,1011	0,1101
mdb019	23,5199	19,9844	10,6191	0,2611	0,2507
mdb020	23,7783	20,9980	9,9749	0,1985	0,2071
mdb021	23,6009	20,9757	10,9282	0,2052	0,2173
mdb022	29,9204	18,6565	9,4157	0,1105	0,1259
mdb023	25,1451	19,6585	9,6070	0,1969	0,2326
mdb024	24,9015	20,5131	10,0738	0,1992	0,2023
mdb025	25,1451	21,1946	10,0831	0,1948	0,2264
mdb026	26,3369	21,8458	10,0654	0,1651	0,1831
mdb027	25,3779	19,4128	9,6941	0,1810	0,1959
mdb028	26,7654	19,4744	10,6515	0,2008	0,1955
mdb029	22,3078	23,6018	10,9914	0,2332	0,2602
mdb030	29,4912	19,2464	9,1348	0,1053	0,1145
average	25,3122	20,3485	10,0960	0,1853	0,1969

Table 7.43: Quadtree time results, th=2, min block size=2, max block size=16

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb017	8,1357	70,9018	20,0402	0,5683	0,6052
mdb018	8,0544	68,9052	20,2031	0,5147	0,5719
mdb015	5,9722	84,0379	25,4283	0,8741	0,8160
mdb027	4,0329	108,0327	33,1151	0,8816	0,9884
mdb024	4,7569	96,3305	31,2784	0,7834	0,8630
mdb028	4,0361	107,4844	33,0585	0,8150	0,9236
mdb013	5,0739	93,7798	27,2871	0,9060	0,9513
mdb022	5,4275	86,8944	28,9360	0,6780	0,8421
mdb023	4,2650	104,0204	31,5932	0,9909	1,1135
mdb030	5,3729	89,8801	25,8601	0,6516	0,7398
mdb026	4,0882	109,3701	31,6166	0,8766	0,9723
mdb016	5,3016	93,3594	28,0626	0,8170	0,8116
mdb025	3,7419	115,1059	33,6527	1,0222	1,1412
mdb014	4,5887	101,1880	30,2180	1,1570	0,9264
mdb021	4,6129	99,3157	28,9380	0,9826	0,9637
mdb011	5,2175	97,0056	26,9853	0,9734	1,0061
mdb019	3,6193	123,7546	34,5160	1,2616	1,3516
mdb029	5,0978	89,2199	27,0433	0,8692	0,9805
mdb020	3,8379	113,0744	34,9868	0,9351	1,0562
mdb012	4,4802	104,1060	30,2028	1,1077	1,2701
average	4,9857	97,7883	29,1511	0,8833	0,9447

Table 7.44: Quadtree time results, th=5, min block size=2, max block size=16

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb017	12,9459	51,2788	16,6586	0,3613	0,3950
mdb018	12,6824	51,7944	15,7970	0,3944	0,4043
mdb015	9,5312	58,8699	18,2513	0,4691	0,5141
mdb022	9,1291	63,1542	20,9580	0,4151	0,4696
mdb016	8,9679	62,3269	21,3958	0,4968	0,5562
mdb030	10,8231	56,8490	18,1996	0,4010	0,4445
mdb024	10,3622	59,6175	20,0850	0,4829	0,6798
mdb013	10,8345	56,1063	17,9015	0,4597	0,5060
mdb028	9,5444	58,1480	19,9404	0,4503	0,5250
mdb011	10,6881	55,9880	17,0401	0,4837	0,5275
mdb021	7,9701	65,5744	21,7004	0,6209	0,6883
mdb026	8,0215	65,9638	21,6968	0,5710	0,7035
mdb014	10,9179	54,0942	17,2083	0,4055	0,4579
mdb027	9,8156	57,7282	18,3058	0,5181	0,5290
mdb023	9,5800	60,8583	19,8149	0,5811	0,6247
mdb025	7,2429	70,9550	22,7612	0,6303	0,6736
mdb029	10,5374	57,8696	19,8880	0,4934	0,6114
mdb020	7,6032	66,4730	22,1870	0,6195	0,6663
mdb019	6,8303	69,9715	22,7409	0,7022	0,7420
mdb012	9,2900	57,0516	18,4181	0,5491	0,6041
average	9,6659	60,0336	19,5474	0,5053	0,5661

Table 7.45: Quadtree time results, th=10, min block size=2, max block size=16

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb017	26,3706	37,1654	11,5534	0,2343	0,2678
mdb018	26,8879	37,0656	11,7144	0,2040	0,2159
mdb015	22,1277	39,9954	12,3287	0,3011	0,3028
mdb022	25,0379	36,5132	11,7848	0,2909	0,2505
mdb016	22,6145	39,3169	12,2291	0,2735	0,3036
mdb024	27,6101	37,3523	11,4924	0,2617	0,3226
mdb013	26,6096	38,6240	12,4492	0,3151	0,3477
mdb030	28,1199	38,6048	12,2191	0,2161	0,2439
mdb011	25,1759	38,9874	13,1545	0,3359	0,3616
mdb014	27,3348	38,9272	12,1760	0,2751	0,2994
mdb027	27,3894	36,2991	12,0201	0,2981	0,3220
mdb012	21,3910	41,9692	13,4000	0,3507	0,3709
mdb028	26,1194	39,1774	12,5019	0,2905	0,2906
mdb029	22,9810	42,9661	13,1912	0,3775	0,3865
mdb023	24,4572	38,3455	12,0063	0,3343	0,3440
mdb021	19,9292	39,1341	12,8290	0,4043	0,3402
mdb026	22,9810	36,9170	12,1176	0,2884	0,3201
mdb025	20,6602	38,5192	12,5241	0,3114	0,3356
mdb020	20,8065	39,3660	12,4556	0,2924	0,3192
mdb019	18,5786	38,9342	13,2391	0,3465	0,3800
average	24,1591	38,7090	12,3693	0,3001	0,3162

Table 7.46: Quadtree time results, th=15, min block size=2, max block size=16

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb017	39,3750	25,3548	8,3442	0,1076	0,1155
mdb018	42,2039	24,0785	8,2094	0,0882	0,0903
mdb011	34,9864	36,4457	11,2891	0,2876	0,2931
mdb022	45,8254	24,7391	8,2728	0,0729	0,0867
mdb024	40,7404	25,6837	9,3983	0,1297	0,1509
mdb013	40,6197	35,1864	10,8473	0,1978	0,2172
mdb015	38,3111	36,5776	11,0377	0,1912	0,2101
mdb012	28,8498	36,7503	11,6511	0,2452	0,2667
mdb016	45,0719	29,5134	8,5050	0,0983	0,1374
mdb030	44,2474	24,7224	8,5107	0,0891	0,1003
mdb014	41,3134	35,2952	11,0939	0,2673	0,2419
mdb023	36,8685	26,0991	9,9939	0,1453	0,1602
mdb021	32,8090	26,1417	9,8579	0,1887	0,1689
mdb027	45,6727	24,3757	8,4913	0,1108	0,1219
mdb029	32,1422	26,4482	9,1558	0,1534	0,1882
mdb028	44,2951	24,5433	8,7112	0,1127	0,1096
mdb026	44,8263	24,4037	8,5478	0,0992	0,1253
mdb020	34,7792	25,8680	9,2257	0,1553	0,1475
mdb025	38,4904	27,1333	8,9953	0,1451	0,1451
mdb019	31,6719	24,7686	9,0729	0,1680	0,1687
average	39,1550	28,2064	9,4606	0,1527	0,1623

Table 7.47: Quadtree time results, th=20, min block size=2, max block size=16

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb017	47,5750	24,6530	8,2775	0,0937	0,1006
mdb018	58,0527	23,3710	7,9728	0,0530	0,0549
mdb011	39,7942	25,3385	8,6747	0,1382	0,1557
mdb013	48,7596	24,6261	8,6269	0,1237	0,1239
mdb012	32,4979	26,2955	9,0606	0,1416	0,2106
mdb029	38,1689	27,3538	9,6022	0,1535	0,1628
mdb024	48,0194	24,3637	8,4987	0,1232	0,1182
mdb023	43,5908	25,4881	8,5132	0,1233	0,1308
mdb015	49,4635	24,8725	8,5422	0,1220	0,1336
mdb030	67,5956	25,3206	7,8998	0,0498	0,0546
mdb014	48,8755	25,5442	8,7061	0,1173	0,1059
mdb022	76,2434	23,3346	8,1544	0,0615	0,0620
mdb027	56,7703	25,8779	8,0336	0,1005	0,1031
mdb016	65,9689	24,0041	7,8216	0,0658	0,0744
mdb028	61,4352	23,9185	8,0756	0,0849	0,0788
mdb021	42,1174	25,8823	8,4659	0,1266	0,1325
mdb020	42,9982	24,2874	8,5881	0,1229	0,1401
mdb026	60,0886	24,8770	8,5202	0,0894	0,0911
mdb025	51,4437	24,8292	8,5983	0,1334	0,1181
mdb019	40,7808	25,5753	8,3568	0,1347	0,1405
average	51,0120	24,9907	8,4495	0,1080	0,1146

Table 7.48: Quadtree time results, th=30, min block size=2, max block size=16

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb018	75,4046	21,5154	7,1439	0,0353	0,0385
mdb017	63,0362	22,0465	7,2160	0,0678	0,0712
mdb011	45,4204	24,3223	8,5238	0,1237	0,1143
mdb013	57,0064	21,3145	6,9290	0,1061	0,0939
mdb029	44,2474	26,0284	8,5977	0,1089	0,1168
mdb014	57,6457	21,2754	6,9327	0,0804	0,0842
mdb012	35,7157	24,2012	7,9864	0,1090	0,1192
mdb015	71,4727	21,8492	7,3282	0,0680	0,0738
mdb024	59,9135	23,4260	7,3557	0,0679	0,0770
mdb030	99,9691	22,1358	6,8947	0,0283	0,0276
mdb027	64,1174	24,0437	7,4923	0,0727	0,0752
mdb016	83,1840	21,7600	7,0269	0,0584	0,0664
mdb023	59,8263	24,4197	7,5481	0,0825	0,0926
mdb022	107,8338	22,5429	6,9750	0,0180	0,0221
mdb021	50,4961	25,2199	8,8174	0,1086	0,1301
mdb028	74,9920	21,3113	6,9897	0,0631	0,0534
mdb020	52,3607	24,4228	8,4268	0,1048	0,1038
mdb026	70,7350	22,8753	8,0884	0,0689	0,1008
mdb025	61,5271	23,5980	7,4474	0,1067	0,1143
mdb019	50,0048	25,3160	8,6153	0,1439	0,1197
average	64,2455	23,1812	7,6168	0,0812	0,0847

Table 7.49: Quadtree time results, th=2, min block size=4, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	13,8252	32,2289	16,9641	0,6362	0,5128
mdb012	12,5931	45,3627	16,6976	0,3234	0,4769
mdb013	13,5384	39,1441	16,1045	0,4010	0,4812
mdb014	12,5126	44,6101	16,8006	0,3403	0,3844
mdb015	15,2732	37,3314	15,2328	0,3718	0,4516
mdb016	14,3461	39,3915	16,5728	0,4231	0,4667
mdb017	17,6307	34,0135	14,2084	0,2902	0,3016
mdb018	17,6231	33,3559	14,1807	0,2290	0,2639
mdb019	11,1974	47,7806	17,5116	0,3078	0,3363
mdb020	11,6610	44,4809	17,3377	0,3831	0,4242
mdb021	12,7730	41,4924	17,0176	0,3256	0,3725
mdb022	14,5491	37,4761	15,3979	0,2619	0,2743
mdb023	11,9839	45,3124	17,0676	0,3522	0,3974
mdb024	12,8009	42,4072	16,6655	0,3515	0,3139
mdb025	11,1791	46,6804	17,1837	0,3324	0,3526
mdb026	11,7880	43,4259	16,8135	0,3405	0,3282
mdb027	11,2341	44,6377	18,7187	0,4269	0,3775
mdb028	11,2896	45,9657	17,4930	0,2390	0,2889
mdb029	13,4675	40,3979	15,4422	0,3686	0,4312
mdb030	14,1877	39,0994	15,3925	0,2675	0,2864
average	13,2727	41,2297	16,4401	0,3486	0,3761

Table 7.50: Quadtree time results, th=5, min block size=4, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	16,0419	36,2734	14,4811	0,3920	0,3693
mdb012	14,2714	38,9967	15,1370	0,3764	0,4343
mdb013	15,5917	36,5239	15,0996	0,3608	0,4049
mdb014	15,2846	36,7892	14,9164	0,3198	0,3380
mdb015	15,9795	34,8419	14,5176	0,3173	0,4118
mdb016	15,4164	36,1057	14,7893	0,4008	0,4392
mdb017	18,5870	33,7843	13,6785	0,2691	0,2990
mdb018	18,5618	32,5999	13,8153	0,2637	0,2719
mdb019	12,4936	42,3483	16,5757	0,3668	0,4157
mdb020	13,1741	40,5393	16,3427	0,2971	0,3351
mdb021	14,3862	38,1650	15,7119	0,3119	0,3656
mdb022	15,5387	37,2182	15,0518	0,2312	0,2558
mdb023	15,0386	37,9388	15,2487	0,2981	0,3412
mdb024	15,4337	36,3680	14,7902	0,2360	0,2748
mdb025	12,8932	39,8623	16,3708	0,3021	0,3377
mdb026	13,8345	40,5416	17,3442	0,2802	0,3208
mdb027	14,6008	37,8142	15,5196	0,3494	0,3770
mdb028	14,8218	38,0068	15,1946	0,2525	0,2831
mdb029	15,9609	35,9158	14,4102	0,3351	0,3831
mdb030	15,8685	34,9438	14,6765	0,2440	0,4165
average	15,1889	37,2789	15,1836	0,3102	0,3537

Table 7.51: Quadtree time results, th=10, min block size=4, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	21,9271	29,8791	13,8958	0,3607	0,4337
mdb012	20,9230	30,4835	13,1154	0,3200	0,3377
mdb013	22,2474	28,9225	12,7413	0,3419	0,3772
mdb014	22,6269	29,2142	12,7478	0,3075	0,3474
mdb015	20,6706	29,8230	13,1196	0,3453	0,3749
mdb016	20,3534	30,5850	13,5815	0,2997	0,3967
mdb017	22,5031	29,6156	13,9200	0,2954	0,2919
mdb018	22,4785	28,5448	12,6036	0,2153	0,2414
mdb019	19,1051	32,2404	14,1299	0,3739	0,4124
mdb020	20,1046	30,1419	13,0402	0,3292	0,3257
mdb021	19,7001	30,5997	13,3683	0,2966	0,3272
mdb022	20,6291	29,6119	13,1980	0,2699	0,2770
mdb023	21,2363	32,1768	12,7588	0,3101	0,3493
mdb024	22,4662	29,6522	13,1019	0,2714	0,2842
mdb025	20,4445	30,9662	12,9878	0,3794	0,3434
mdb026	20,8065	29,8694	13,1348	0,2952	0,3384
mdb027	22,6893	27,7771	12,8192	0,3289	0,3533
mdb028	22,4662	29,7435	12,9960	0,2687	0,2957
mdb029	21,7416	30,6723	12,7977	0,2840	0,3250
mdb030	22,6768	29,0186	12,7751	0,2285	0,2918
average	21,3898	29,9769	13,1416	0,3061	0,3362

Table 7.52: Quadtree time results, th=15, min block size=4, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	25,2998	27,6985	12,1606	0,2997	0,3418
mdb012	24,3126	27,1807	12,3751	0,3153	0,3163
mdb013	25,7433	27,1030	12,1893	0,3099	0,3813
mdb014	26,0532	26,4157	11,9570	0,2418	0,2845
mdb015	25,1913	28,4627	12,3003	0,3536	0,3693
mdb016	25,7111	26,8301	12,4751	0,2272	0,2450
mdb017	25,6470	27,6203	12,4665	0,2311	0,2573
mdb018	25,5513	27,2596	12,7017	0,1819	0,2033
mdb019	24,3126	28,1706	13,1486	0,4062	0,4469
mdb020	25,0991	28,2267	12,6120	0,3057	0,3501
mdb021	24,7964	28,5616	13,5287	0,3162	0,4214
mdb022	25,8729	27,7569	14,0045	0,2334	0,2426
mdb023	25,5513	28,8175	13,0712	0,3180	0,3324
mdb024	26,3706	28,4854	12,7003	0,2970	0,2680
mdb025	25,4564	29,3033	12,9707	0,2764	0,3144
mdb026	26,3200	27,3225	12,6150	0,2475	0,2998
mdb027	26,8353	27,0449	12,3438	0,2599	0,2902
mdb028	26,2193	27,3942	12,7537	0,2196	0,2432
mdb029	24,3126	28,5644	12,9300	0,3194	0,3529
mdb030	25,9545	28,1456	12,0043	0,2455	0,2315
average	25,5305	27,8182	12,6654	0,2803	0,3096

Table 7.53: Quadtree time results, th=20, min block size=4, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	26,4725	27,4362	11,9369	0,3171	0,3523
mdb012	25,3466	27,6351	12,5849	0,2867	0,2828
mdb013	27,3166	27,3259	12,7566	0,3675	0,3608
mdb014	27,1006	27,4772	12,1156	0,2636	0,2965
mdb015	26,7829	27,9488	12,8087	0,3080	0,2987
mdb016	28,1007	26,2142	12,8920	0,2451	0,2751
mdb017	26,5752	27,2012	12,2658	0,3252	0,2532
mdb018	26,7480	26,1772	13,1985	0,1751	0,2089
mdb019	26,3200	27,7285	12,8949	0,3343	0,3713
mdb020	26,8003	26,3814	12,6853	0,2483	0,2803
mdb021	26,4896	27,3508	13,1538	0,3022	0,3611
mdb022	28,8904	26,2782	12,1043	0,2338	0,2393
mdb023	26,8353	26,4004	11,5612	0,2399	0,2686
mdb024	27,3894	24,8242	11,6617	0,2582	0,2539
mdb025	27,3894	26,1706	11,6171	0,2439	0,2653
mdb026	28,0623	24,7793	11,3699	0,2387	0,2548
mdb027	28,0241	24,9212	11,6105	0,2392	0,2636
mdb028	28,0432	24,6969	11,5308	0,1942	0,2186
mdb029	25,9709	25,5430	11,8143	0,2906	0,3163
mdb030	27,9859	26,2211	11,4630	0,1698	0,1937
average	27,1322	26,4356	12,2013	0,2641	0,2808

Table 7.54: Quadtree time results, th=30, min block size=4, max block size=8

name	Compression Rate	encoding time	decoding time	aritmetic encoding time	aritmetic decoding time
mdb011	26,9232	25,3162	11,6340	0,2906	0,2805
mdb012	25,7756	25,5900	11,7968	0,2351	0,2628
mdb013	27,9479	24,6794	11,9827	0,2923	0,3279
mdb014	27,9669	25,6325	11,5197	0,2966	0,3198
mdb015	28,6488	25,7052	11,5518	0,2796	0,3038
mdb016	29,3230	25,7618	11,3426	0,2075	0,2304
mdb017	27,6844	24,7612	11,4268	0,2342	0,2385
mdb018	27,7218	25,2018	11,5540	0,1754	0,1734
mdb019	27,4259	25,2713	12,3390	0,2767	0,3257
mdb020	27,4993	27,6718	12,4871	0,2534	0,2741
mdb021	27,4259	26,7068	13,5044	0,2875	0,3334
mdb022	29,9859	26,6996	11,9012	0,2273	0,2168
mdb023	28,1007	26,2754	11,9577	0,2895	0,2899
mdb024	28,0623	27,0590	11,7871	0,1970	0,2208
mdb025	28,2357	25,3538	12,1443	0,2392	0,2561
mdb026	28,6888	25,2670	11,7674	0,2263	0,2438
mdb027	28,3330	26,1887	11,9897	0,2456	0,2874
mdb028	28,9107	25,6579	12,0578	0,1891	0,2121
mdb029	26,8353	27,2081	12,6635	0,2889	0,3145
mdb030	29,5124	27,4175	11,7419	0,1943	0,2143
average	28,0504	25,9713	11,9575	0,2463	0,2663

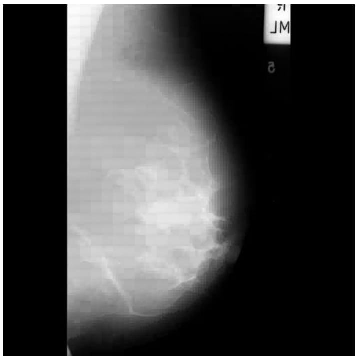


Figure 7.19: mdb014 Quadtree based reconstructed image,max block size=16,min block size=2, th=10, CR=40,315812, PSNR=43,801308, MSE=0,000093, ssim=0,956746

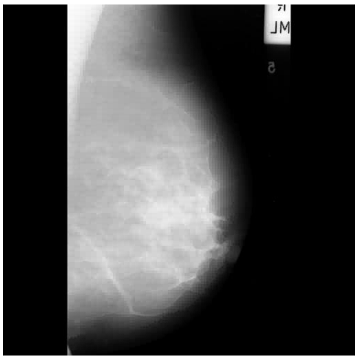


Figure 7.20: mdb014 Quadtree based reconstructed image, max block size=8, min block size=2, th=10, CR=18,503357, PSNR=41,885735, MSE=0,000065, ssim=0,965658

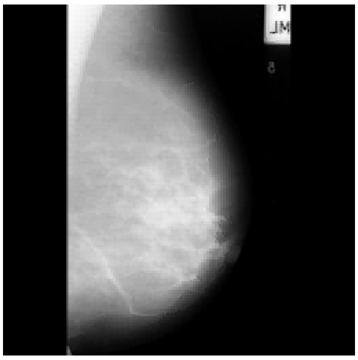


Figure 7.21: mdb014 Quadtree based reconstructed image, max block size=8, min block size=4, th=10, CR=27,100589, PSNR=31,111927, MSE=0,000774, ssim=0,952584

Figure 7.22: Block size effect on Quadtree base reconstruct image at th=10

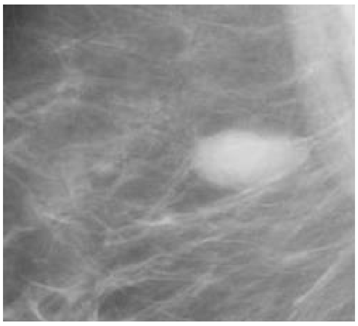


Figure 7.23: Main mdb025 Image, size 256×256

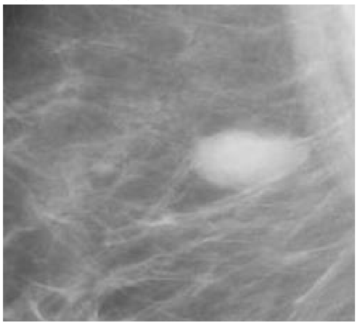


Figure 7.24: mdb025 Reconstruct image by 2×2 Block size, CR=1,882353, PSNR=47,6, MSE=0,000017, ssim=0,991013

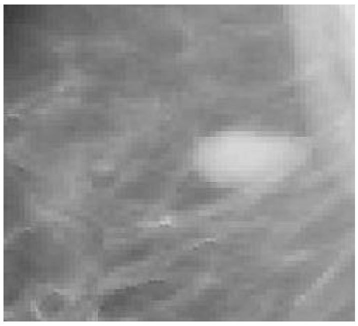


Figure 7.25: mdb025 Reconstruct image by 4×4 Block size, CR=7,529412, PSNR=34,7654, MSE=0,000334, ssim=0,933879

Figure 7.26: mdb025 reconstructed image by 8×8 block size, CR=30,117647, PSNR=31,364375, MSE=0,000227, ssim=0,936326

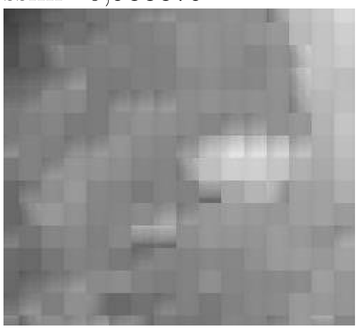


Figure 7.27: mdb025 reconstructed image by 16×16 block size, CR=120,470688, PSNR=31,4839, MSE=0,000711, ssim=0,8756

Figure 7.28: Effect of increasing block size

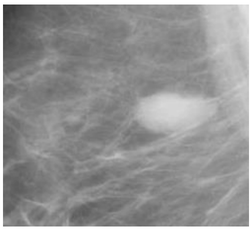


Figure 7.29: mdb014 Quadtree based reconstructed image,max block size=16,min block size=2, th=2, CR=3,741867, PSNR=47,483093, MSE=0,000018, ssim=0,989665

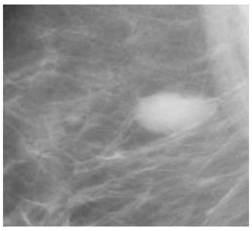


Figure 7.30: mdb025 Quadtree based reconstructed image, max block size=8, min block size=2, th=2, CR=3,615212, PSNR=47,405836, MSE=0,000018, ssim=0,989941

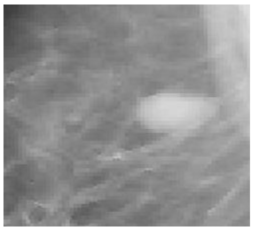


Figure 7.31: mdb025 Quadtree based reconstructed image, max block size=8, min block size=4, th=2, CR=11,179147, PSNR=33,0104, MSE=0,0005, ssim=0,932125

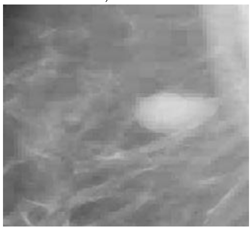


Figure 7.32: mdb014 Quadtree based reconstructed image,max block size=16,min block size=2, th=10, CR=20,660171, PSNR=39,437701, MSE=0,000114, ssim=0,937867

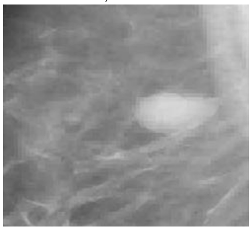


Figure 7.33: mdb025 Quadtree based reconstructed image, max block size=8, min block size=2, th=10, CR=16,073209, PSNR=40,30966, MSE=0,000093, ssim=0,947031

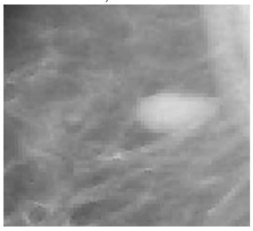


Figure 7.34: mdb025 Quadtree based reconstructed image, max block size=8, min block size=4, th=10, CR=20,444462, PSNR=32,989134, MSE=0,000502, ssim=0,932002

Figure 7.35: mdb014 Quadtree based reconstructed image,max block size=16,min block size=2, th=20, CR=51,443654, PSNR=35,395715, MSE=0,000289, ssim=0,897822

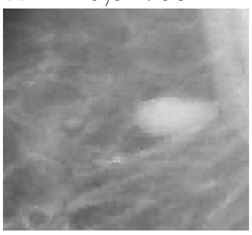


Figure 7.36: mdb025 Quadtree based reconstructed image, max block size=8, min block size=2, th=20, CR=24,240889, PSNR=39,411749, MSE=0,000115, ssim=0,941273

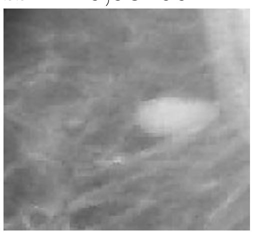


Figure 7.37: mdb025 Quadtree based reconstructed image, max block size=8, min block size=4, th=20, CR=27,389405, PSNR=33,01774, MSE=0,000499, ssim=0,933205

Figure 7.38: Affect of image compression based on Quadtree method, on breast fibroadenoma image(zoom part) by different thresholds

Chapter 8

Conclusion

In this paper, a new Mammography Compression Method Based on Classied Energy and Pattern Building Block (CEPB) sets is proposed. In the method, first the CEB and CPB sets are constructed and any image data can be reconstructed block by block using a block scaling coefficient and the index numbers of the classified energy and pattern blocks placed in the CEB and CPB.

The CEB and CPB sets are constructed for different sizes of image blocks such as 2 by 2, 8 by 8 or 4 by 4 with respect to different compression ratios desired.

Reconstructing of Image has been done based on 2 different method.

- fix block size Encoding and decoding
- Quadtree based encoding and decoding

the medical images have to be very accurate and details in Mammography image are related to image quality sensitively. The satisfying level of the distortions for medical images was mentioned in Chapter4.

At the end of a series of the experimental works, the evaluation results show that the proposed method provides high compression ratios while preserving the image quality . When the compression ratio versus image quality. Moreover, Results by Quadtree shows better compression ratio than fixed size block method. The experiments proved that the proposed method can obtain the level ($PSNR = 30 - 47dB$). Furthermore, an image of this quality is compressed at compression ratio about ($120 : 1, 1, 8 : 1$) for fixed block size. By increasing block sizes Compression ratio, arithmetic compression ratio and MSE increase, and PSNR, SSIM decrease. The best results in this part has been seen in $(4, 8)$ block sizes. The results for different images have near values to each other. Also, as time values that checked in this method encoding, decoding, arithmetic encoding and arithmetic decoding times has been measured too. Again, we have same near values too. The most used time is related to encoding part that the average values change between 117-27s. Average decoding time changes between 27-0,4 s for $(2, 16)$ block sizes. One of the most important issues about this thesis is the showing different aspects of images values that all details about image quality and time information in this process is available and has been shown.

In the other hand, as Quadtree method it shows the huge increase in PSNR and Compression rate quality. Thus, the use of Quadtree method reduces the file size more that the first step. Moreover, this method is about two times efficiently than the lossless compression methods specially in low thresholds for quadtree (compression ratios not larger than $4 : 1$). At compression ratios from 4,9 $: 1$ to $24 : 1$ the PSNR varies from 40 dB to 47 dB for 2-16 block size. One of the most important aspects of thesis is searching both threshold and block sizes affect on image compression.
All values for image compression in threshold values by 2, 5, 10, 15, 20, and 30 has been shown for

- min block size = 2, max block size=8,
- min block size = 2, max block size=16,
- min block size = 2, max block size=4.

The average compression value for

- min block size = 2, max block size=8 is about 17.04
- min block size = 2, max block size=16 is about 24,16
- min block size = 2, max block size=4 is 21,38.

PSNR values changes minimally in this method and near to each other. for example in

- min block size = 2, max block size=8 in threshold=10 is 41,46
- min block size = 2, max block size=16 is 40,4
- min block size = 2, max block size=4 is 45,53

SSIM and MSE values change have been shown too.
Time results has been shown in this part too. we have decreasing in this values specially in encoding time that is higher in fixed block size method.

The accuracy of method is high as quadtree method has been chosen and most aspects of quadtree and the impact of it in image has been shown for different max and min block size in quadtree and the affect of Threshold increasing in different block size has been shown too. By this way it would be so easy to choose the best Threshold and max and min block sizes in future tasks. It should be mention that mammography images for different patient are so near to each other and important details are very small to sense in images. This method by this quality can perform very important role in medical image compression, as details that can be seen in mammography is not visible in ultrasound or MRI images. The good compression can play important role to used these images easily in hospitals PACS system to compare them.

For the time being, the performance of the newly proposed method is measured using PSNR, MSE, SSIM metrics that SSIM part is newer.

In our future works we will be focused on better designed CEB and CPB in order to increase the level of the PSNR while reducing the number of bits required representing the image blocks. Moreover, the methods that can search on better performance as time issues aspect . In the other hand it would be great to search about methods that can apply as post process after decoding in order to image quality as enhancement such as using filtering method as Savitzky-Golay filter.

References

[1] W. R. E. Gonzalez, R. C., *Digital image processing.* New Jersey,2nd edn, ISBN 0-201-18075-8: Prentic-Hall, 2002.

[2] M. J. Dallwitz, "An introduction to computer images,," *TDWG Newsletter*, vol. 7, no. 10, pp. 1–3, 1992.

[3] K. E. Anderson-Lemieux, A., "Digital image resolution: what it means and how it can work for you.. (German)," *Professional Communication Conference, 1999. IPCC 99.Communication Jazz: Improvising the New International Communication Culture. Proceedings.*, pp. 231–236, 1999.

[4] R. P. Starosolski, "metod bezstratnej kompresji obrazw medycznych."

[5] S. Shapiro, "Evidence on screening for breast cancer from a randomized trial,," *Cancer*, vol. 39, 1977.

[6] (2015) Mammography - canadian cancer society. [Online]. Available: http://www.cancer.ca/en/cancer-information/diagnosisand-treatment/tests-and-procedures/mammography/?region=on

[7] " Breast Imaging Reporting and Data System,," *2013 ACR BI-RADS Atlas,American College of Radiology,*, 2013.

[8] W. JN., "Breast Parenchymal Patterns And Their Changes With Age,," *Radiology (3 Pt. 1),*, pp. 545–552, 1976.

[9] S. o. P. D. Deorowicz, *Universal lossless data compression algorithms*, Silesian University of Technology, 2003.

[10] A. K. d. Przelaskowski, "Knuth: Computers and typesetting,." [Online]. Available: http://www.ire.pw.edu.pl/~arturp/Dydaktyka/koda/skrypt.html

[11] W. Walczak, "Fractal compression of medical images." 2008.

[12] S. H. Koff, D. A., "An overview of digital compression of medical images: can we use lossy image compression in radiology?. journal =."

[13] B. R. Oh, T. H., "Jpeg2000 and jpeg: image quality measures of compressed medical images.. journal =."

[14] B. B. Mandelbrot, *The fractal geometry of nature.* New York: W. H. Freedman and Co., 1983.

[15] R. M. S. D. Hartenstein, H., "Region-based fractal image compression,," *IEEE Transactions on Image Processing*, vol. 9(7), pp. 1171–1184, 2000.

[16] D. J.-L. Polidori, E., "Zooming using iterated function systems. Fractals,," *IEEE Transactions on Image Processing*, vol. 5, pp. 111–123, 2001.

[17] J. E. W. B. R. D. Fisher, Y., *Fractal image compression using iterated transforms. W Image and text compression, (J. A. Storer, ed.), roz. 2, pp. 3561.* Norwell, MA, USA, and Dordrecht,The Netherlands, 1992. ISBN 0-7923-9243-4: Kluwer Academic Publishers Group, 1993.

[18] Y. Fisher, *Fractal image compression*, ser. 92 COURSE NOTES. Raport tech. 12, Department of Mathematics, Technion Israel Institute of Technology, 1992.

[19] M. Novak, "Attractor coding of images." *W Proceedings of the International Picture Coding Symposium PCS93, Lausanne*, 1993.

[20] d. J. G. Wohlberg, B., "A review of the fractal image coding literature." *IEEE Transactions on Image Processing,*, vol. 8(12), no. 10, pp. 1716–1729, 1999.

[21] Y. C. Wu, X., "mage coding by adaptive tree-structured segmentation," *IEEE Transactions on Information Theory*, pp. 73–82, 1991.

[22] E. Reusens, "Partitioning complexity issue for iterated function systems based image coding," *Proceedings of the VIIth European Signal Processing Conference EUSIPCO94*, vol. I, pp. 171–174, 1994, Edinburgh.

[23] S. J. C. J.-M. Davoine, F., "A mixed triangular and quadrilateral partition for fractal image coding." *W Proceedings of IEEE International Conference on Image Processing ICIP-95*, vol. III, pp. 284–287, 1995.

[24] C. J.-M. Davoine, F., "daptive delaunay triangulation for attractor image coding." *W Proc. of 12th International Conference on Pattern Recognition (ICPR)*, pp. 801–803, Jerusalem, 1994.

[25] A. M.-C. J.-M. B. M. Davoine, F., "Fractal image compression based on delaunay triangulation and vector quantization." *IEEE Transactions on Image Processing*, vol. 5(2), pp. 338–346, 1996.

[26] D. F. Thomas, L., "Region-based fractal image compression using heuristic search." *IEEE Transactions on Image Processing,*.

[27] R. M. Saupe, D., "Evolutionary fractal image compression." *W Proc. ICIP-96 IEEE International Conference on Image Processing, Lausanne, Switzerland,*, vol. I, pp. 129 –132, 1996.

[28] O. H.-K. T. Tanimoto, M., "A new fractal image coding scheme employing blocks of variable shapes." *W Proceedings ICIP-96 (IEEE International Conference on Image Processing), Lausanne, Switzerland*, vol. 1, pp. 137–140, 1996.

[29] H. H.-S. D. Ruhl, M., "Adaptive partitionings for fractal image compression.." *Proceedings ICIP-97 (IEEE International Conference on Image Processing), anta Barbara, CA, USA*, vol. II, pp. 310–313, 1997.

[30] S.-B.-K.-W. J.-S. Chang, Y.-C., "Region-based fractal image compression with quadtree segmentation." *W ICASSP 97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 97)*, vol. 4, no. 10, p. 3125, 1997.

[31] S. B.-W.-J. Chang, Y.C., "Region-based fractal compression for still image." *W In Proc. WSCG20000, The 8-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media, Plzen, Czech Republic*, 2000.

[32] S. D. COchotta, T., "Edge-based partition coding for fractal image compression.." *The Arabian Journal for Science and Engineering, Special Issue on Fractal and Wavelet Methods,*, vol. 29(2c), 2004.

[33] S. R. Tate, *Lossless compression of region edge maps.*, ser. Raport tech. Technical report DUKETR199209. Duke University, Durham, NC, USA, 1992.

[34] J. E. W.-B.-R. D. Fisher, Y., *Fractal image compression using iterated transforms*, ser. W Image and text compression, (J. A. Storer, ed.), roz. 2. Kluwer Academic Publishers Group, Norwell, MA, USA, and Dordrecht, The Netherlands, 1992.

[35] M. S. Fisher, Y., "Fractal encoding with hv partitions." pp. 119–126, 1995.

[36] G. J. H.-T.-S. T. Frigaard, C., *Image compression based on a fractal theory*, ser. Internal Report S701,. Institute for Electronic Systems, Aalborg University, Aalborg, Denmark, 1994.

[37] A. E. Jacquin, "Fractal image coding." *a review. Proceedings of the IEEE,*, vol. 81(10), pp. 1451 –1465, 1993.

[38] M. D. M. Woolley, S. J., "optimum parameters for hybrid fractal image coding,," *W Proceedings of ICASSP-1995 IEEE International Conference on Acoustics, Speech and Signal Processing, Detroit,*, 1995.

[39] S. C. Hurtgen, B., "Fast hierarchical codebook search for fractal coding of still images.." *Proceedings of the SPIE The International Society for Optical Engineering*, vol. 1977, pp. 397 –408, 1993.

[40] V. e. T. Barthel, K. U., "Adaptive fractal image coding in the frequency domain." *W Proceedings of the International Workshop on Image Processing*, vol. XLV, pp. 33 –38, 1994.

[41] B. D. M. D. M. Wakefield, P., "Hybrid image compression with implicit fractal terms." *W ICASSP 97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 97)*, vol. 4, 1997.

[42] D. F. Monro, D. M., "Fractal approximation of image blocks." *W Proceedings of ICASSP-1992 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 458 – 488, 1992.

[43] D. M. Monro, "Class of fractal transforms." *Electronics Letters*, vol. 29(4), pp. 362 – 362, 1993.

[44] A. E. Jacquin, "A novel fractal block-coding technique for digital images." *Proceed- Bibliography 166 ings of IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP*, vol. 4, pp. 2225–2228, 1990.

[45] J. M. Beaumont, "Advances in block based fractal coding of still pictures." *W Proceedings IEE Colloquium: The application of fractal techniques in image processing,*, pp. 3.1–3.6, 1990.

[46] W. S. J. Monro, D. M., "Rate/distortion in fractal compression." *order of transform and block symmetries. W Proc. ISSIPNN94 Intern. Symp. on Speech, Image Processing and Neural Networks*, vol. 1, pp. 168–171, 1994-Hong Kong.

[47] N. Lu, "Fractal imaging. Academic Press." 1997.

[48] H. Kaouri, "Fractal coding of still images." *W IEE 6th International Conference on Digital Processing of Signals in Communications*, pp. 235–239, 1991.

[49] R. M. Saupe, D., "Evolutionary fractal image compression." *IEEE International Conference on Image Processing, Lausanne, Switzerland*, vol. I, pp. 129–132, 1996.

[50] M. D. M. Woolley, S. J., "Rate-distortion performance of fractal transforms for image compression,," *Fractals*, vol. 2(3), pp. 395–398, 1994.

[51] B. Z. L. S. K.-E. M. D. ien, G. E., "A new improved collage theorem with applications to multiresolution fractal image coding.." *W Proceedings ICASSP-94 (IEEE International Conference on Acoustics, Speech and Signal Processing),Adelaide, Australia*, vol. 5, pp. 565–568, 1994.

[52] G. E. l. ien, "optimal attractor image coding with fast decoder convergence." *Doctor of Philosophy Dissertation, The Norwegian Institute of Technology.*, 1993.

[53] L. S. R. T. A. ien, G. E., "An inner product space approach to image coding by contractive transformations,," *W Proceedings of ICASSP-1991 IEEE International Conference on Acoustics, Speech and Signal Processing,*, pp. 2773–2776, 1991.

[54] D. M. Monro, "energialized fractal transforms,," *W Proceedings DCC93 Data Compression Conference, (J. A. Storer, M. Cohn, eds.)*, pp. 254–261, 1993.

[55] H. T. S. Gharavi-Alkhansari, M., "Generalized image coding using fractal-based methods." *W Proceedings of the International Picture Coding Symposium PCS94, Sacramento, California, 1994.*, pp. 440–443, 1994.

[56] ——, "Generalized image coding using fractal-based methods." *W Proceedings of the International Picture Coding Symposium PCS94, Sacramento, California, 1994.*, pp. 440–443, 1994.

[57] ——, "Fractal image coding using rate-distortion optimized matching pursuit." *W Proceedings of SPIE Visual Communications and Image Processing 1996 (R. Ansari, M. J. Smith, eds.)*, vol. 2727, pp. 1386–1393, 1996.

[58] ——, "Ffractal-based image block-coding algorithm." *W Proceedings of ICASSP-1993 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, pp. 345–348, 1993.

[59] G. Vines, "Orthogonal basis ifs."

[60] K. W. K. K. K. P. Kwiatkowski, J., "Using fractal coding in medical image magnification." *W PPAM, (R. Wyrzykowski, J. Dongarra, M. Paprzycki, J. Wasniewski, eds.), Lecture Notes in Computer Science*, vol. 2328, pp. 517–525, 2001.

[61] H. Lin, *Fractal image compression using pyramids.* Doctor of Philosophy Dissertation, University of Toronto., 1997.

[62] Y. B. Zhao, Y., "A new affine transformation: its theory and application to image coding." *IEEE Transactions on Circuits and Systems for Video Technology,*.

[63] K. S.-D. Kang, H.-S., "Fractal decoding algorithm for fast convergence." *Optical Engineering*, vol. 35(11), pp. 3191–3198, 1996.

[64] R. Hamzaoui, "Ordered decoding algorithm for fractal image compression," *W Proceedings of the International Picture Coding Symposium PCS97*, pp. 91–95, 1997.

[65] M. D. K.-E. Baharav, Z., "ierarchical interpretation of fractal image coding and its applications to fast decoding." *W Intl. Conf. on Digital Signal Processing, Cyprus,*, 1993.

[66] ——, "ierarchical interpretation of fractal image coding and its applications.w fisher [fis95a]."

[67] D. F. Monro, D. M., "Rendering algorithms for deterministic fractals." *IEEE Compututer Graphics and Applications,*.

[68] A. MZakhor, "Iterative procedures for reduction of blocking effects in transform image coding." *IEEE Transactions on Circuits and Systems for Video Technology,*.

[69] T. G. Breazu, M., "based fractal image compression using deterministic search." *W Proc. ICIP-98 IEEE International Conference on Image Processing, Chicago,*, 1998.

[70] K. S. Fu and K. J. Mui, "A survey on Image Segmentation Through Clustering," vol. 13, no. 1, pp. 3– 16, 1981.

[71] J. Ashburner and K.J.Friston, "Image Segmentation,," vol. 15, pp. 433–438, 2008.

[72] N. Rajini and R. Bhavani, "Enhancing k-means and kernelized fuzzy cmeans clustering with cluster center initialization in segmenting mri brain images," *Electronic Computer Technology (ICECT),2011 3rd International Conference*, 2011.

[73] W. Y. Q. Zhang, SA. Goldman and J. Fritts, "Content based image retrival using multiple instance learning," pp. 682–689, 2002.

[74] Z. Y. W. Li and X. Shixiong, "A Novel Clustering Algorithm Based on Hierarchical and K-Means Clustering." *Control Conference, 2007, Chinese*, pp. 605–609, 2009.

[75] S. S. N.Isa and U. Ngah, "Adaptive Fuzzy Moving K-Means Clustring Algorithm for Image Segmentation,," *Consumer Electronics,IEEE Tansactions*, vol. 55, pp. 2145–2153, 2009.

[76] "Learning, pp.21-26," 2002. [Online]. Available: http://www.learningace.com/cs525/2002-spring-cs525-lecture-2

[77] U. Guz, "A Novel Image Compression Method Based on Classified Energy and Pattern Building Blocks,," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, 2011.

[78] A. N. Akansu and R. A. Haddad, *Multiresolution Signal Decomposition*. San Diego, Calif, USA: Academic Press, 1992.

[79] J. G. Proakis and D. K. Manolakis, *DigitalSignalProcessing*. New York, NY, USA, 4th edition: Prentice Hall, 2006.

[80] M. Gezer, "Original Mathematical Methods in Image Coding and Compression in Modern Communication Systems,," *Istanbul university, PHD Thesis*, 2014.

[81] S. T. Welstead, *Fractal and wavelet image compression techniques*. SPIE Publication, 1999.

[82] Lehmann, E. L.; Casella, George, *Theory of Point Estimation (2nd ed.)*. Springer, 1998.

[83] K. Pearson, "On lines and planes of closest fit to systems of points in space]," *Philosophical Magazine*, vol. 6, no. 2, pp. 559–572, 1901.

[84] *Pixel grids, bit rate and compression ratio*. Clarendon Press, 2007-12-01. Retrieved 2013-06-05.

[85] A. S. H. S. E. Wang, Zhou; Bovik, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing. 13 (4)*, vol. 322, no. 10, pp. 600–612, 2004-04-01.

[86] D. J. MacKay, *Chapter 6: Stream Codes". Information Theory, Inference, and Learning Algorithms (PDF/PostScript/DjVu/LaTeX)*, ser. Archived from the original on 22 December 2007. Retrieved 2007-12-30. Cambridge University Press, 2003.