# CODEC FOR MULTIMEDIA SERVICES USING WAVELETS AND FRACTALS

by

**Yarish Brijmohan**
**BScEng (Electronic)** ***Summa cum Laude***

Submitted in fulfilment of the requirements for the
Degree of Master of Science in Electronic Engineering
in the School of Electrical, Electronic and Computer Engineering
at the University of KwaZulu-Natal, Durban

November 2004

# Preface

The research described in this dissertation was performed at the University of KwaZulu-Natal (Howard College Campus), Durban, over the period March 2003 until October 2004 by Mr. Yarish Brijmohan under the supervision of Professor Stanley Mneney.

This work has been generously sponsored by Armscor and Thales Advanced Engineering.

I hereby declare that all the material incorporated in this dissertation is my own original unaided work except where specific acknowledgment is made by name or in the form of a reference. The work contained herein has not been submitted in whole or part for a degree at any other university.

Signed : 

Name : Mr. Yarish Brijmohan

Date : 10 November 2004

# Acknowledgements

First and foremost, I would like to express my appreciation to God and Her Holiness Shri Mataji Nirmala Devi for inspiration, guidance and support. All work conducted by myself, is in credit to the Divine.

I wish to thank my supervisor, Professor Stanley Mneney, for his deep enthusiasm, guidance, encouragement and supervision during the course of this research. His comments throughout were constructive and insightful.

I would also like to express my sincere thanks to my family for their everlasting support, encouragement and invaluable assistance throughout my life.

Furthermore, I express my appreciation to all staff and fellow post-graduate students who have assisted me in any way and for the exciting non-work related discussions and activities.

Finally, special thanks go out to the sponsor's Armscor and Thales Advanced Engineering for the funding of the DSP kit.

## Publications

The following publications are based on the work presented in this dissertation.

Y. Brijmohan and S.H. Mneney, "*Benchmark for Modern Wavelet Based Still Image Compression*," South African Telecommunications and Networking Applications Conference (SATNAC), George, South Africa, Sept. 2003.

Y. Brijmohan and S.H. Mneney, "*Modern Wavelet and Fractal Based Image Coding*," Military Information and Communications Symposium of South Africa (MICSSA), Pretoria, South Africa, Nov. 2003.

Y. Brijmohan and S.H. Mneney, "*Video Compression for Very Low Bit-Rate Communications using Fractal and Wavelet Techniques*," South African Telecommunications and Networking Applications Conference (SATNAC), Stellenbosch, South Africa, Sept. 2004.

Y. Brijmohan and S.H. Mneney, "*Low Bit-Rate Video Coding using Fractal Compression of Wavelet Subtrees*," 7th IEEE Africon Conference, Gaborone, Botswana, Vol. 1, pp. 39 - 44, Sept. 2004.

# Abstract

Increase in technological advancements in fields of telecommunications, computers and television have prompted the need to exchange video, image and audio files between people. Transmission of such files finds numerous multimedia applications such as, internet multimedia, video conferencing, videophone, etc. However, the transmission and reception of these files are limited by the available bandwidth as well as storage capacities of systems. Thus there is a need to develop compression systems, such that required multimedia applications can operate within these limited capacities.

This dissertation presents two well established coding approaches that are used in modern image and video compression systems. These are the wavelet and fractal methods. The wavelet based coder, which adopts the transform coding paradigm, performs the discrete wavelet transform on an image before any compression algorithms are implemented. The wavelet transform provides good energy compaction and decorrelating properties that make it suited for compression. Fractal compression systems on the other hand differ from the traditional transform coders. These algorithms are based on the theory of iterated function systems and take advantage of local self-similarities present in images. In this dissertation, we first review the theoretical foundations of both wavelet and fractal coders. Thereafter we evaluate different wavelet and fractal based compression algorithms, and assess the strengths and weakness in each case.

Due to the short-comings of fractal based compression schemes, such as the tiling effect appearing in reconstructed images, a wavelet based analysis of fractal image compression is presented. This is the link that produces fractal coding in the wavelet domain, and presents a hybrid coding scheme called fractal-wavelet coders. We show that by using smooth wavelet basis in computing the wavelet transform, the tiling effect of fractal systems can be removed. The few wavelet-fractal coders that have been proposed in literature are discussed, showing advantages over the traditional fractal coders.

This dissertation will present a new low-bit rate video compression system that is based on fractal coding in the wavelet domain. This coder makes use of the advantages of both the wavelet and fractal coders discussed in their review. The self-similarity property of fractal coders exploits the high spatial and temporal correlation between video frames. Thus the fractal coding step gives an approximate representation of the coded frame, while the wavelet technique adds detail to the frame. In this proposed scheme, each frame is decomposed using the pyramidal multi-resolution wavelet transform. Thereafter a motion detection operation is

used in which the subtrees are partitioned into motion and non-motion subtrees. The non-motion subtrees are easily coded by a binary decision, whereas the moving ones are coded using the combination of the wavelet SPIHT and fractal variable subtree size coding scheme. All intra-frame compression is performed using the SPIHT compression algorithm and inter-frame using the fractal-wavelet method described above.

The proposed coder is then compared to current low bit-rate video coding standards such as the H.263+ and MPEG-4 coders through analysis and simulations. Results show that the proposed coder is competitive with the current standards, with a performance improvement been shown in video sequences that do not posses large global motion. Finally, a real-time implementation of the proposed algorithm is performed on a digital signal processor. This illustrates the suitability of the proposed coder being applied to numerous multimedia applications.

# Table of Contents

# List of Figures

# List of Tables

# List of Acronyms

bpp : bits per pixel
kbit/s : kilobits per second
Mbit/s : Megabits per second
fps : frames per second
MSE : Mean Square Error
PSNR : Peak Signal to Noise Ratio
R-D : Rate-Distortion
RGB : Red, Green, Blue
YCrCb : Luminance, Chrominance Red, Chrominance Blue
CIF : Common Intermediate Format
QCIF : Quadrature Common Intermediate Format
ME : Motion Estimation
MC : Motion Compensation
MD : Motion Detection
2-D : Two Dimensional
3-D : Three Dimensional

IFS : Iterated Function Systems
CMP : Contraction Mapping Principle
PIFS : Partitioned Iterated Function Systems
FBC : Fractal Block Coder
IFSM : Iterated Function Systems on Grey Level Maps
RB : Range Block
DB : Domain Block

DCT : Discrete Cosine Transform
DWT : Discrete Wavelet Transform
FT : Fourier Transform
MRA : Multi-resolution Analysis
FWT : Fast Wavelet Transform
EZW : Embedded Zerotree Wavelet
SPIHT : Set Partitioning in Hierarchical Trees
LSP : List of Significant Pixels
LIP : List of Insignificant Pixels

LIS : List of Insignificant Sets

SR : Stack-Run

ECECOW: Embedded Conditional Entropy Coding of Wavelet Coefficients

SFQ : Space Frequency Quantization

PPC : Predictive Pyramid Coder

Av/s : Average and Sub-sampling Operator

SQS : Self-Quantization of Subtrees

WFC : Wavelet Fractal Coder

FZW : Fractal Zerotree Wavelet

DSP : Digital Signal Processor

IDK : Imaging Developer's Kit

DMA : Direct Memory Access

ISO : International Standards Organisation

JPEG : Joint Photographic Experts Group

MPEG : Motion Picture Experts Group

ROI : Regions of Interest

MB : Macro-Block

GOB : Group of Blocks

GOP : Group of Pictures

EOF : End of File

[a,b) : Interval from a to b, including a but not b

# CHAPTER 1 - INTRODUCTION

The development of image and video capturing devices has revolutionised the multimedia industry. Multimedia developers have recognized the importance of using images and video as a valuable means of information capturing and communication. As the old saying goes, "A picture speaks louder than a thousand words". Hence, complex processes or events can be represented with greater accuracy and efficiency by using a video clip, rather than a textual description of the process.

As faster and more efficient technological advancements occurred, there has been an increasing need to exchange video and image files among users from different parts of the worlds or between people from similar organizations. This has prompted a development of a unified representation of video and image signals. Currently, there are numerous image and video formats that exist in industry today. What is lacking is a system that incorporates different image and video formats and provides a standard representation to enable user interoperability. Two development groups were formed by the International Standards Organization (ISO) to address this very issue. These groups are the JPEG (Joint Photographic Experts Group), which deals with still image related issues, and the MPEG (Moving Picture Experts Group), that addresses video matters. These groups developed standards on both image and video such that they provided a common language to facilitate the communication between different parties. Their job was also to develop systems that efficiently represent and store the media.

If the media can be stored using less bandwidth than required by the transmission channel, then successful communications of media can be obtained. This subsequently gives rise to numerous multimedia applications. Specifically, if the media is represented by a low bit-rate (few number of bits), applications such as video-conferencing, internet multimedia, videophone and wireless multimedia can be implemented. This reduction in the bit-rate is achieved by the development of data compression algorithms that reduce the number of bits required to represent an image or a video sequence. The questions that arise are how can data compression be achieved? And how much compression can be attained? To answer these questions we need to consider and understand the characteristics of images and video sequences and how they are used in defining a compression process. The key point to remember is that the process must produce a representation of the media that is acceptable to the average human observer.

## 1.1 Basics of Image and Video Coding

### 1.1.1 Images

An image can be thought of as a positive function defined onto a plane, which consists of different colours or shades of grey. The value of the function at each point in a greyscale image defines the luminance or brightness of the image at that point. For colour pictures, the functional representation is extended to a three dimensional plane. There are different representations of the colour image, such as the RGB and YCrCb formats. To construct digital images, each function is sampled at discrete locations in each plane. The sample values are known as pixels. Each pixel value is represented to a predefined precision, called number of bits per pixels (bpp). Hence a greyscale digital image forms a matrix I of size $N$ by $M$, where each pixel in I at a discrete location (x,y), represents the luminance value at the point represented with a predefined number of bits. Colour digital images produce an image matrix I which has dimensions of $N$ by $M$ by 3. Each pixel value in the colour image consists of three components that represent the intensity at that point.

#### 1.1.1.1 Colour Spaces

a) RGB (Red, Green, Blue) – This is the simplest and most commonly used colour space, also known as a true colour image. Each pixel is represented by a proportion of the red, green and blue component. These components are normally scaled to a maximum and minimum value. Most colours in the visible spectra can be recreated using a proportional combination of these components [56].

b) YCrCb – This format is widely used in digital video. In this space, the image is represented as a single luminance ($Y$) space, and two colour difference chrominance spaces, namely the chrominance red ($Cr$) and chrominance blue ($Cb$). $Cr$ represents the difference between the red component and a reference value, while $Cb$ is the difference between the blue component and a reference [68]. A conversion between the RGB and YCrCb space can be obtained by the following equations:

$$\begin{aligned} Y &= 0.299\,R + 0.587\,G + 0.114\,B \\ Cr &= ((B-Y)/2) + 0.5 \\ Cb &= ((R-Y)/1.6) + 0.5 \end{aligned} \qquad (1.1)$$

The advantage of using the YCrCb format is that there is no correlation between the individual spaces and hence each space can be analysed separately. Thus image operations performed on greyscale images (the luminance component in colour images) can easily be extended to colour images. Therefore, throughout this dissertation we do not explicitly treat colour images. Another

advantage of the YCrCb format is that human eyes are more sensitive to brightness than colour. Hence the *Cr* and *Cb* spaces can be compressed higher than the *Y* space, to achieve a better overall compression.

#### 1.1.1.2 Image Coding

The main problem of representing images with formats detailed in the previous section is that the size in bytes to store an image is extremely large. For example, suppose that the resolution of the greyscale image is 512x512 and each pixel's precision is 8 bpp. The storage requirement for this image would be $512^2 \times 8 = 2\,097\,152$ bits or 262 144 bytes. If we wanted to transmit this image over a typical transmission line, such as 56 kbit/s modem, it would take approximately 38 seconds. Consider several such images being transmitted. Using the previous calculation, it will take too much time to transmit this sequence of images. A solution to this problem would be to increase the capacity (or bandwidth) of the transmitting channel. However, this comes at a greater cost. Hence a solution to the limited bandwidth would be to reduce the storage size of the image. This solution is known as image compression, where the image is represented in a compact form.

Compression algorithms perform the task of reducing the number of bits used to represent an image or an image sequence. Hence these algorithms form a mapping of the image into strings of bits. For each of the compression algorithms, there must exist an inverse process (or decompression algorithm), in which the original image can be recovered. The amount of compression acquired is the ratio of the size of the original media to the size of the compressed media. This is better known as the compression ratio. There are several compression algorithms existing today that achieve different compression ratios for different types of images. These algorithms fall into two broad categories; lossless compression and lossy compression. Lossless compression occurs when the original image is recovered exactly after decompression. This is normally achieved by entropy coders, which yield a much shorter image on an average by using short code words for likely images and longer code words for unlikely images.

Most of the time exact reconstruction of the image is not a necessity. Therefore one can introduce a small loss of data in the image, without having noticeable effects. This is known as lossy compression and achieves a much higher compression ratio than lossless compression schemes. The greater the loss introduced, the higher the compression ratio can be. The limit placed on the amount of loss incurred, is based on the average human observer. Generally, there must not be too much of a blurring effect on the reconstructed image and the image must be acceptable to the observer. Several measures have been defined to quantify the acceptableness

of a compressed image, with the most successful being the peak signal to noise ratio (PSNR) measure.

Sampled images are represented in the spatial-time domain, which creates a huge volume of data with high spatial redundancy. Thus in this form the image is unsuitable for compression. To overcome the redundancy, the image must be transformed to another domain, giving rise to transform based encoders. The majority of lossy compression algorithms are based on transform coders.

### 1.1.2 Video

A video is a sequence of images which are displayed in a particular order, to create the illusion of motion. The rate at which these images (or frames) are displayed is known the frame-rate and is measured in frames per second (fps). Typically, the frame rate is 30 fps to perceive continuous motion. Like images, video sequences also have different formats. Video formats are defined in terms of the number of pixels per line, the number of lines per picture and the pixel aspect ratio [67]. The pixel aspect ratio is the ratio of pixels per line to number of lines. Table 1-1 defines standard video sequence formats. These formats are scaled versions of the CIF (Common Intermediate Format). Each pixel in these formats is represented by the YCrCb image format. As alluded to earlier in Section 1.1.1.1, human observes are less sensitive to the chrominance components. Hence the resolution of the chrominance components of each frame can be reduced to half the resolution of the luminance component. Hence Table 1-1 depicts the resolution of the *Y* component, whereas the resolution *Cr* and *Cb* components are half the size of the *Y* in both the horizontal and vertical directions. This is known as the 4:2:0 format (sometimes called 4:1:1) [67].

| | Sub-QCIF | QCIF | CIF | 4CIF |
|---|---|---|---|---|
| No. of Pixels per Line | 128 | 176 | 352 | 704 |
| No. of Lines | 96 | 144 | 288 | 576 |
| Uncompressed Bit-rate (at 30 fps) | 4.4 Mbit/s | 9.1 Mbit/s | 37 Mbit/s | 146 Mbit/s |
| Pixel Aspect Ratio | 4:3 | 11:9 | 11:9 | 11:9 |

Table 1-1: Standard Video picture formats [67]

#### 1.1.2.1 Video Coding

Table 1-1 also details the enormous amount of data contained in each of the full motion video sequences. It is clear that some kind of compression needs to be implemented. There are several options that can be used to reduce the size of the video, including reducing the frame-rate, reducing the frame resolution, or reducing the precision of the intensity. Implementing these options come at a price. For example, reduction in frame rate removes the feeling of continuous motion of the sequence, reduction in resolution reduces the picture clarity and reducing the number of bits per pixel (or precision) would make pictures look unnatural. Hence the only option left is to apply some sort of a compression algorithm that accounts for the properties of the video.

a) Intra/Inter Frame Coders

Compression is normally achieved by trying to eliminate redundancy in the video data. There are two types of redundancies that are present in natural video, namely spatial redundancy and temporal redundancy. Spatial redundancy refers to the correlation that is present between different parts of a frame [67]. Coding of frames consisting of spatial redundancy is identical to image coding. This type of coding is known as intra-frame coding, which involves coding of a frame by itself using image compression algorithms.

Temporal redundancies, on the other hand are redundancies present between frames [67]. It is noticed that if the frame rate is sufficiently high, successive frames in the video sequence, contain very similar information. Exploiting these redundancies, frames are coded using information from previous frames. This is known as inter-frame coding. Inter-frame coding is normally performed using a Motion Estimation (ME) and Motion Compensation (MC) approach. ME, as the name says, find moving objects which causes the slight difference in consecutive frames. Thereafter, MC is used to predict the motion structure, using data from previous frames. Since there is not much difference between consecutive frames, large amount of data that is repeated in the current frame need not be coded. This consequently removes the temporal redundancy.

In intra/inter frame coders, the first frame in the sequence is always intra coded, for obvious reasons. In decompressing the sequence, if a part of the intra-frame is lost, this error will be rippled though the sequence until the next intra-frame is reached. Thus it is vital in a large sequence to perform intra coding at a regular rate.

b) Three Dimensional (3-D) Coders

Three dimensional techniques perform video coding on groups of frames where each group is coded as a still image. These frames are grouped together to form 3-D structure, called a group of pictures, as depicted in Figure 1-1. 3-D algorithms exploit both spatial self-similarities as well as temporal redundancies. In coding, the group of frames need to be partitioned into cubes in an optimal manner according to their statistical properties. The division can be performed along the temporal or spatial axis.

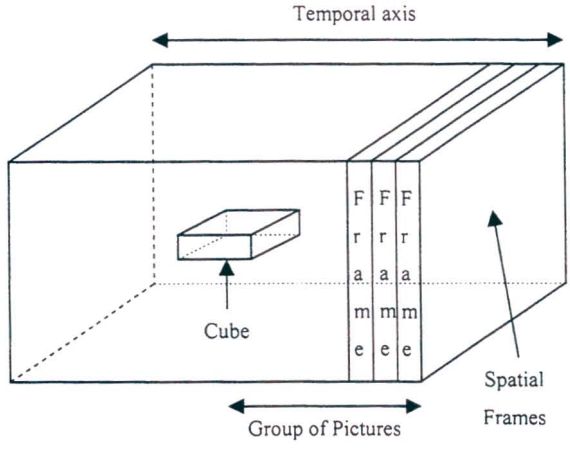



Figure 1-1: Three dimensional structures used for video coding

## 1.2 Wavelets and Fractals

This dissertation concerns itself with compression algorithms that use wavelets, fractals and a combination of both of them. Before wavelets were introduced, signals could either be represented in the time domain or the frequency domain. The time representation provided the exact spatial or temporal information, but shed no light on the spectral (or frequency) information. The Fourier transform was used to compute the spectral information of signals, but represented no temporal data. Thus wavelet theory was developed which bridged this gap, in that it provided a combined time-frequency analysis of signals.

Over the last 20 years, wavelets have found numerous applications in signal processing theory and practice. One of the most important applications is in image and multi-resolution analysis. Multi-resolution theory is concerned with the representation and analysis of images at more than one resolution. The wavelet transform has the ability of providing a multi-resolution and multi-frequency decomposition of images. Wavelet analysis provides images with certain properties that lend itself to image and video compression. One of the most important properties is that the wavelet transform decorrelates mutually dependant parts of the image and performs an energy

compaction of the samples representing the image. Once wavelet analysis had proven successful in compression, image compression standards (such as the JPEG-2000) based their coding algorithms on wavelet ones.

Fractal image compression on the other hand differs from the standard transform coder methods. They were created as a result of the study of iterated function systems (IFS). Fractal compression exploits similarities that are present in different scales throughout the image (also known as self-similarities). Using fractal theory, an image may be represented by a system known as the fractal transform, which consists of IFS. Fractal methods store images as contraction maps of which the image is the fixed point. For natural images, the parameters of the fractal transform can be coded very compactly, which make fractal methods suited for image coding. The decoding procedure is simple, in which the image is recovered by iterating the maps to its fixed point. However, this dissertation will show that the standard fractal compression methodology has not matched the rate-distortion performance of wavelet based techniques.

The combination of wavelet and fractal coding is a relatively new area of research. Fractal coding was generalized from the spatial domain to the wavelet domain, independently by the works of G.M. Davis [28] and H. Krupnik [27]. Davis then introduced a new term, the wavelet subtree, which consists of wavelet coefficients from the same spatial location but with different resolutions and orientations in the wavelet domain. He then demonstrated that using Haar wavelets in the quantization scheme is equivalent to the original fractal block coder. It was also illustrated that by using a smooth wavelet basis in computation of the wavelet transform, the tiling effects present in fractal coders can be dramatically reduced. Thus a combination of fractal and wavelet techniques has the potential of providing good compression systems.

## 1.3 Layout of Dissertation

This dissertation concerns itself with media compression methods based on fractal and wavelet techniques. Thus the first five chapters of this dissertation are dedicated in building up the theoretical ground work of these compression systems. Furthermore, these chapters provide a literature review of the most important media compression algorithms developed. The following two chapters utilize specific properties of the compression algorithms highlighted in the literature review, to propose and assess the performance of a new video compression system.

Chapter 2 provides a general introduction to image and video compression systems. The majority of effort is placed in detailing the transform based compression methodology, which is employed in the majority of media compression algorithms. An important sub-system of the transform based method is the entropy coding stage. Two frequently used entropy coding algorithms, namely the Huffman and Arithmetic coders, are discussed. Also included in this chapter is a review of the different image and video coding standards that are present in industry today. For image compression, the standards looked at are the JPEG and JPEG-2000, while for low bit-rate video compression, the standards include H.263 and MPEG-4. In order to compare different compression schemes, several metrics are defined in this chapter.

Chapter 3 introduces fractals and iterated function systems. The initial portion of this chapter provides an overview of the basic IFS theory and establishes several theorems, such as the contraction mapping principle, that are fundamental in the production of fractals. The inverse problem for IFS is posed and the collage theorem is developed as a solution. Fractal image compression algorithms are based on the collage theorem and these methods take advantage of local self-similarity that is present in natural images. A generic image compression algorithm, based on the Jacquin [5] scheme, is presented in this chapter. Several enhancements to this general coding scheme are developed, which increases the performance of fractal methods. Finally this chapter concludes by detailing different fractal video compression algorithms that exist in literature.

Chapter 4 covers the mathematical properties of wavelets and the wavelet transform. This chapter begins by introducing wavelet bases and the computation of the discrete wavelet transform via Mallat's [42] filter-bank method. After the mathematical framework has been established, several properties of the wavelet transform are detailed, that accounts for its effective application in compression systems. The final section of this chapter presents a literature survey of wavelet image compression algorithms. Each algorithm is discussed, with results presented to assess the performance.

With the theory of fractals and wavelets well established in the previous chapters, Chapter 5 investigates the ability of combining these two compression methods to produce a compression system with superior performance. This chapter starts with a wavelet analysis of the general fractal coding scheme provided in Chapter 3. This involves producing fractal coding in the wavelet domain. Using the results of the above analysis, a basic image compression algorithm was developed that performs fractal compression on the wavelet coefficients. Thereafter, a performance comparison between the newly established wavelet based fractal coder and the

traditional fractal coder is undertaken. Finally, all existing wavelet based fractal image and video compression algorithms are discussed and compared.

Having embarked on this literary and theoretical review of fractal and wavelet compression algorithms, a new video compression system is proposed in Chapter 6. This compression algorithm entails fractal coding in the wavelet domain. This chapter initially presents an outline of the compression system, focussing on the major sub-systems. Thereafter, the entire proposed algorithm is analysed in detail. For each component in the algorithm reasons for choice as well as effectiveness in compression are presented.

Chapter 7 details all results obtained during the complete evaluation of the proposed video compression system. The chapter starts off by defining the platform and testing methods to be used in the evaluation. Thereafter a comparison of the performance of the proposed algorithm against the H.263+ and MPEG-4 compression standards are conducted on different video test sequences characterizing different motion structures. The performance measurements are in terms of amount of compression, recovered video quality and computational complexity. The next part of the chapter analyses the complexity of each sub-component of the proposed system. Finally all results achieved during the testing phase are presented and accounted for.

Chapter 8 provides a real time implementation of the proposed video coder on a digital signal processor. Initially a general description of the DSP platform is provided, detailing the speed and memory specifications of the DSP. Thereafter, several implementation issues for each component of the video system are discussed. For each of these subsystems complexity reduction techniques are supplied. The performance testing of this implementation was executed on different real time video clips, entailing different motion structures. This chapter concludes by listing several multimedia applications that the proposed video coder can be used for.

The last chapter, Chapter 9, summarizes all important points highlighted throughout this dissertation in a methodical manner. A discussion of the strengths and weaknesses of the proposed compression system is presented. Based on this discussion, possible improvements and future research are proposed.

## 1.4 Executive Summary

The main aim of this project is to recommend the best compression codecs that can be utilized to provide different multimedia services. However, these codecs must employ compression algorithms based on fractal and wavelet techniques. Visual multimedia applications can be broken into two main categories, namely still pictures and video. Thus this dissertation will provide the best performing and most efficient compression systems for both of these categories.

This dissertation details and compares all fractal, wavelet and fractal-wavelet image compression systems present in literature. The results indicate that wavelet methods outperform fractal based systems in both compression ratio and recovered image quality. Furthermore, the computational complexity of fractal systems is much greater. The introduction of hybrid fractal-wavelet compression schemes has closed the performance gap to wavelet methods. However, the computational overhead of the hybrid schemes far overweighs the performance gain of these systems. Thus for still image applications, wavelet compression methods should be employed. This dissertation shows that the most effective wavelet-based image compression algorithms are the SPIHT [48] and ECECOW [52] methods.

Fractal methods are much more suited to video coding, since self-similarities are present throughout the video sequence. Results produced in this dissertation illustrates that fractal compression in the wavelet domain removes the blocking artefacts present in almost all low bit-rate video compression algorithms. Therefore, this dissertation proposes a new low bit-rate video compression system that combines fractal and wavelet methods. All coding is performed in the wavelet domain.

The proposed coder is based on the intra/inter frame coding concept. This algorithm initially performs the wavelet transform on each frame of the sequence individually. The first frame is intra coded with the SPIHT image compression algorithm. Coding of successive frames, employs a motion detection operation that separates motion subtrees from non-motion ones. All motion subtrees are coded by an innovative method that combines the algorithms of the variable tree size fractal coder [37] and the SPIHT method. Further reduction in the bit-rate is achieved by including an adaptive arithmetic entropy encoder.

A comprehensive performance evaluation of the proposed system was conducted. The results portrayed that the proposed algorithm produces superior visual quality for different compressed

video sequences when compared to the H.263+ and MPEG-4 compression standards. Furthermore the encoding times are much faster for the proposed method. This new system generates very low-bit rates on sequences that contain few local motion structures. For scenes with global motion, the visual quality of the video is severely reduced to maintain these low bit-rates. Due to the low computational complexity of the proposed system, a real time implementation on a digital signal processor was executed.

# CHAPTER 2 - STANDARD TYPES OF COMPRESSION

There are two types of image (or video) compression namely lossless and lossy. In lossless compression the original image is recovered exactly after decompression. However, it is difficult to obtain error-free compression above a 2:1 rate. Therefore to obtain much higher compression ratios, the compression must be lossy, in that there must exist a small error between the decompressed and original image. Lossy compression can be motivated by the fact that in many cases it is not necessary or desirable to achieve a perfect, error-free reconstruction of the original image. The main factor that determines if the lossy technique is successful is the acceptability factor, which is explained in Section 2.2.

Section 2.1 presents the most commonly used lossy compression system, the transform coder. Each sub-component of this system is analysed. Also in this section, algorithms of the two most commonly used entropy coders are included. Section 2.2 defines all the standard metrics that can be used to compare different compression algorithms. These include measurements for the amount of compression achieved and for the recovered image quality. Section 2.3 introduces image and video compression standards that exist today. Several properties of these standards are presented.

## 2.1 Transform Based Compression Schemes

In general, transform based compression systems consists of three subsystems. These compression schemes first involve the transformation of spatial information to another domain. At the first sub-system, the pixels of the input image undergo a mathematical transformation that rearranges the information content of the original pixels, so that they can be represented compactly in the new transformation space. The main aim of this step is to decorrelate the spatially distributed energy into fewer data samples such that no information is lost and also to remove the redundancy in the transformed image. The transformation must have an inverse transformation to reconstruct the compressed image in the spatial domain, such that the image can be recovered. In general, the transform and the inverse transform stages are lossless. The steps of the transform based encoder and decoder are shown in Figure 2-1. All loss of information occurs at the quantization stage. Quantizing refers to a reduction of the precision of the values of the transform. For compression to be achieved, transform values must be expressed with fewer bits for each value. It is at this point where compression algorithms take

different approaches. Normally, during the quantization process, less visually significant transformed coefficients are discarded. The final stage of the coding process involves the implementation of an entropy coder. Entropy coding minimizes the redundancy in the bit stream by cleverly exploiting whatever repetitive pattern is left in the quantized data. Furthermore, this sub-system is fully invertible at the decoding stage, so it is lossless. The output of the entropy encoder organizes the data bit stream in a manner necessary for the correct operation of the decoding process.

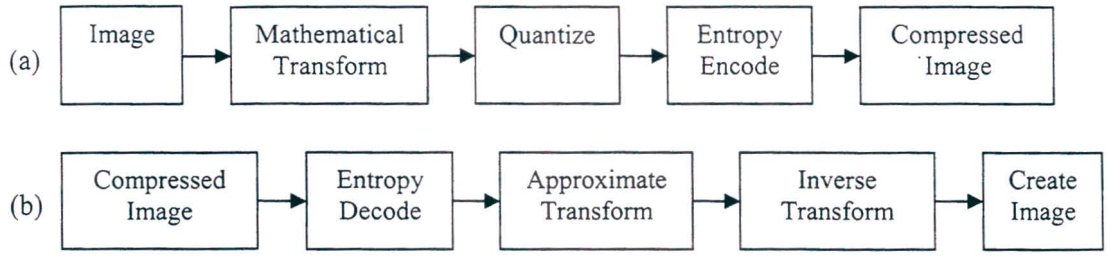



Figure 2-1: Block Diagram of a Transform Based Compression System
(a) Encoding procedure, (b) Decoding procedure

### 2.1.1 Commonly Used Mathematical Transformations

a) Discrete Cosine Transform (DCT) – The cosine transform is a special case of the Fourier Transform (FT), in which the imaginary components of the FT are dropped. The DCT is used in the majority of compression standards developed to date. DCT is normally computed on an 8x8 image block. When the DCT is applied to the image block, the coefficients of the transform are arranged according to the spectral content, i.e. the lower frequency information appear at the upper left hand corner of the transformed block, while the higher frequency data appear at the lower right hand corner. The DCT also concentrates the majority of the energy in the lower frequencies. Hence the coefficients at the upper left hand corner are more significant with respect to those at the lower right, thus making compression easier. The coefficients are normally numbered in a zigzag order (Figure 2-2) from the top left to bottom right so that there will be many smaller coefficients at the end. This is vital for image compression using the DCT. Two dimensional (2-D) DCT requires 54 multiplications, 468 additions and shift operations [67].

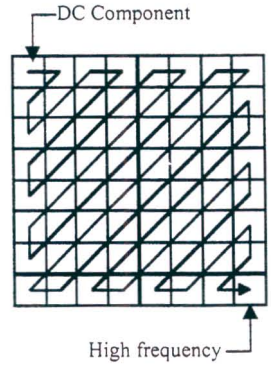



Figure 2-2: Scan order of the DCT coefficients [67]

b) Discrete Wavelet Transform (DWT) – The wavelet transform provided a multi-resolution, multi-frequency representation of a signal. For images, the wavelet transform concentrates the majority of the energy in the lower frequency subbands. The wavelet transform and its properties that lend itself to image compression are detailed in Section 4.2.

### 2.1.2 Entropy Coders

Entropy coding is a lossless coding scheme, in which symbols are coded according to their probabilities of occurrence. The basic idea of these systems is to represent more probable symbols with shorter code-words (or bits) and less probable ones, with longer code-words. The limit placed on the least number of bits to represent a sequence of symbols is given by Shannon's Theorem.

**Theorem 2-1: Shannon's Coding Theorem:** Given a randomly generated message from a source (or model), it is impossible to encode the message into fewer bits (on an average) than the entropy of that source. The entropy is given by

$$H(S) = \sum_{k=1}^{n} -p\{m_k\} \log_2 p\{m_k\}, \tag{2.1}$$

where $S$ is the source that consists of symbols $m_1, m_2, ..., m_n$ and $p\{m_k\}$ is the probability of the source producing the symbol $m_k$.

The probability of each symbol being produced is determined by the model of the system. The model may assign a predetermined probability to each symbol, where the symbol probabilities do not change in the encoding and decoding process. The other alternative is that an adaptive model be used, where the symbol probabilities are modified based on the frequency of

occurrence in the system. Two well known entropy coders, namely the Huffman and Arithmetic coders, are used widely in compression algorithms. These coders are briefly outlined next.

#### 2.1.2.1 Huffman Coder

The Huffman coder is a variable length coding scheme that assigns to each symbol a code-word that corresponds to its probability. The code-word assigned consists of an integer number of bits. Code-words for symbols that have a higher probability (or frequency) of occurring are shorter than the lower probability ones. This causes a reduction in the average code length of a sequence of symbols which provides the necessary compression. The Huffman coder provides an instantaneous codeword for a symbol, in which no other codeword is a prefix of another. This provides efficient decoding.

The Huffman encoding algorithm uses a binary tree structure that holds that symbols at its leaf nodes. This algorithm appears in Algorithm 2-1.

**Algorithm 2-1: Huffman Encoder**

1. List source symbols in decreasing order of probability of occurrence.
2. Combine the least probable symbol pair to produce a new reduced source, with the probability being the sum of the pair probabilities.
3. Label the lines connecting the symbol pair to the new reduced symbol as '0' and '1'.
4. Rearrange the symbols of the reduced source in decreasing order of their respective probability and repeat Step 2 until a single combined symbol of probability 1 remains.
5. The Huffman code for the original source symbols is given by the sequence of '0's and '1's on the branches from the final reduced symbol to the original source symbol.

**Example 2-1:** Suppose the source S produces symbols m = {A, B, C, Z} with probabilities p = {0.5, 0.3, 0.1, 0.1}. The Huffman coding tree is formed as follows:

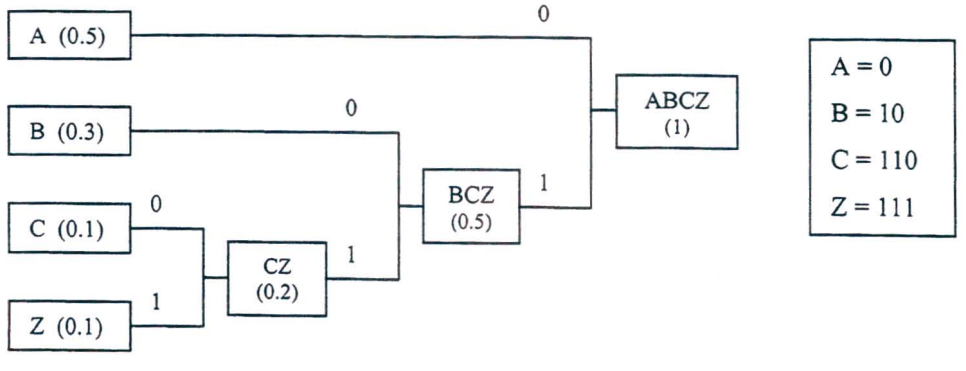



Figure 2-3: Huffman Encoding of the sequence in Example 2-1

In the above example, it is noticed that the code-words assigned to each symbol is not unique. We could interchange the 0 and 1 at each branch, but maintaining consistency. The entropy of this system is given by 1.685 bits/symbol. The average code length is calculated by multiplying the probability of each symbol by its code length. For the above example, the code length is 1.7 bits/symbol. It can therefore be seen that the average code length achieved by the Huffman coder is greater than the entropy of the system, but better than 2.00 bits/symbol required if no coding was used. Further information of Huffman codes can be explored in [63].

In the Huffman coder, the optimum number of bits to be used for each symbol is $\log_2(1/p\{m_k\})$. Thus if the probability of a symbol is really small, then the number of bits used to represent it becomes very high. Another problem with this coding scheme is that the code to represent a symbol must be an integer. Hence if the optimal number of bits is a fraction, then the number of bits is rounded up. One can clearly see that this is a sub-optimal approach which implies that the probabilities of symbols in the Huffman coder must be an integral power of half, which is not always the case [62]. Another downfall with the Huffman method is that it is very difficult to implement adaptive schemes.

#### 2.1.2.2 Arithmetic Coder

The arithmetic coder removes the restrictions that accounts for the downfalls of the Huffman coder, in that in its coding scheme, it removes the idea of replacing each symbol with a codeword. Instead, the arithmetic coder assigns a codeword to a set of symbols. The codeword is normally in the form of a floating point number that is in the interval of 0 and 1. The basic idea of the arithmetic coder is to modify the interval of 0 to 1 according to the probabilities of each symbol in the data set [60]. As the number of symbols in the set increases, the interval needed to represent it becomes smaller and the number of bits to specify this interval increases. Symbols with a higher probability reduce the interval less than that of a lesser probability. The encoding algorithm is depicted in Algorithm 2-2. Each data set is normally terminated by an end of file (EOF) symbol that indicates to the decoder that the end of the data set has been reached. It is shown in [60] that the decoder can uniquely decode the data set if any number in the final interval is specified.

**Algorithm 2-2: Arithmetic Encoder:**

1. Initialise the current interval [L, H) to [0, 1).
2. For each symbol in the data set, do
   a. Subdivide the current interval into subintervals according to the probability of each symbol.
   b. Select the interval corresponding to current symbol and make a new interval.
3. Output enough bits to distinguish the final current interval from all other intervals.

**Example 2-2:** Suppose that a source S produces symbols m = {A, B, C, Z} with symbol probabilities p = {0.5, 0.3, 0.1, 0.1}. Let the data set be composed of the following sequence {A, C, A, B, Z}, with Z being the EOF symbol. Proceeding with Algorithm 2-2, we obtain:

- I = [0, 1)
- Subdivision: $I = [0,0.5) \cup [0.5,0.8) \cup [0.8,0.9) \cup [0.9,1.0)$
- Input symbol = A, so I = [0, 0.5)
- Subdivision: $I = [0,0.25) \cup [0.25,0.4) \cup [0.4,0.45) \cup [0.45,0.5)$
- Input symbol = C, so I = [0.4, 0.45)
- Subdivision: $I = [0.4,0.425) \cup [0.425,0.44) \cup [0.44,0.445) \cup [0.445,0.45)$
- Input symbol = A, so I = [0.4, 0.425)
- Subdivision: $I = [0.4,0.4125) \cup [0.4125,0.42) \cup [0.42,0.4225) \cup [0.4225,0.425)$
- Input symbol = B, so I = [0.4125, 0.42)
- Subdivision: I =

  $[0.4125,0.41625) \cup [0.41625,0.4185) \cup [0.4185,0.41925) \cup [0.41925,0.42)$
- Input symbol = Z, so I = [0.41925, 0.42)
- Pick any value in I, e.g. 0.4193

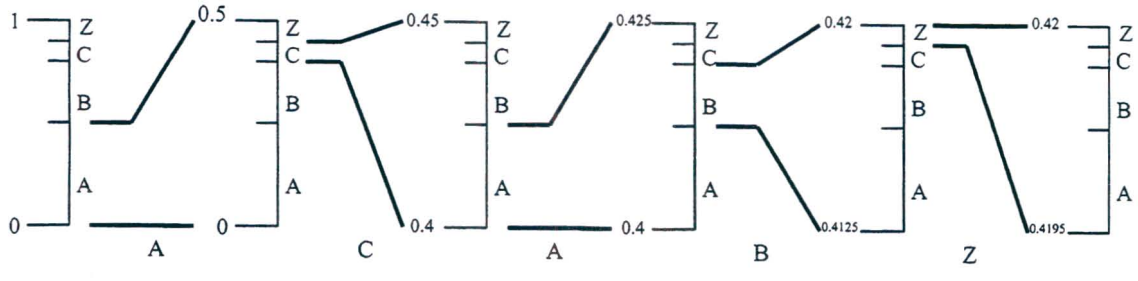



Figure 2-4: Arithmetic Encoding of the sequence in Example 2-2

The above process is graphically illustrated in Figure 2-4, where at each stage the interval is zoomed in. The entropy of the above system is 3.12 (using base 10 since the encoding was performed in decimal). Thus the sequence {A,C,A,B,Z} can be represented by 4 decimal digits.

To implement the arithmetic coding algorithm on a computer, it was found useful to perform integer arithmetic. In practice, the initial interval is specified to be between some large integer number and zero. Using the algorithm described in Algorithm 2-2, the decoder can only start decoding once the entire data set is encoded. Therefore it was found necessary to implement an incremental transmission, which output bits every time a symbol is coded. Furthermore an adaptive model is used in which the symbol probability is updated on each occurrence. All the finer details and algorithms can be found in [60].

## 2.2 Measuring Performance

Once a compression algorithm is developed we need to compare its performance to other algorithms. Hence there must be some sort of a metric that can be used to evaluate performance. Generally, we are interested in two aspects of the compression algorithm, namely, the amount of compression and the quality of the result. For real time implementations of any compression algorithm, other factors such as compression (or decompression) times and memory requirements become critical.

Two commonly used metrics have been defined to indicate the amount of compression an algorithm can achieve. These are the compression ratio, and the bit-rate.

**Definition 2-1: Compression Ratio**: The compression ratio can be defined as a measure of the ratio between the original data size to the compressed data size. This can be restated as follows:

$$\text{Compression Ratio (CR)} = \frac{\text{number of bits in original image}}{\text{number of bits in compressed image}}. \qquad (2.2)$$

For example, if a 512x512, 8-bit greyscale image (2097152 bits) is reduced to 16384 bits, then the image has been compressed to a ratio of 128:1.

**Definition 2-2: Bit-Rate**: The bit-rate is defined as the average number of bits used to represent each pixel in the image, i.e.

$$\text{bit-rate} = \frac{\text{number of bits in compressed image}}{\text{number of pixels}}. \qquad (2.3)$$

The bit-rate is measured in bits per pixel (bpp). For video sequences, the bit-rate is defined as the number of bits output per second and is measured in kbit/s.

For the above example, the number of bits in the compressed image is 16384 and the number of pixels is $512^2$. So the bit-rate of the compression algorithm will be 0.0625 bpp. It must be noted that high compression ratios correspond to low bit-rates.

The quality of compression is much more difficult to quantify, since the perceived quality of the image is dependant on the viewer and its usage. What is required is a measure of the inaccuracy of the approximation between the original image and the recovered image after compression. One approach will be to use the distance function to define the error (or distortion). This distance function is termed the mean square error.

**Definition 2-3: Mean Square Error (MSE)**: The MSE between two images A and B is defined as:

$$MSE = \frac{1}{N \times M} \sum_{i,j} [A(i,j) - B(i,j)]^2 , \tag{2.4}$$

where NxM is the total number of pixels in each image, and sum is over all pixels in the image.

The values of the MSE are somewhat inconvenient to compare, hence another metric, the peak signal to noise ratio, is defined to address this problem.

**Definition 2-4: Peak Signal to Noise Ratio (PSNR)**: The PSNR is defined in terms of the MSE. The PSNR between two images is given as:

$$PSNR = 10 \log_{10} \left( \frac{\left[ 2^{(bit-rate\ of\ original\ image)} - 1 \right]^2}{MSE} \right) \text{dB}. \tag{2.5}$$

The unit of the PSNR is the decibel (dB).

In practice, higher PSNR indicate that the distortion between the original and recovered images is smaller, in the human observer sense. Generally, if the PSNR is 40dB or larger, the two images are virtually indistinguishable by average human observers. The PSNR also has some flaws. For example, if each row of the recovered image is shifted one pixel to the right (translations), the image features will fair poorly in terms of PSNR, but not necessarily for the viewer. However, despite its flaws, the PSNR is still universally used as a measure of image quality.

To compare the performance between different compression algorithms, it is convenient to use rate/distortion (R-D) curves. The x-axis in the R-D curve represents the bit-rate (or compression ratio), while the y-axis depicts the distortion measurement, such as the PSNR.

## 2.3 Image & Video Compression Standards

Advances in many aspects and applications of digital technology, such as imaging and video, have prompted the development of equipment that supports the required application. As explained in Section 1.1.2, storing and transmitting uncompressed raw video is not a good idea, since it requires large storage space and huge bandwidth to transmit. Algorithms have been developed that can compress image and video so that the quality is not degraded to an unacceptable level. These algorithms take into account and utilize special characteristics of the media in order to yield a compression ratio. In order to enable interoperability of equipment from different manufacturers, standards in image and video compression needs to be introduced. This benefits customers in that they will have a greater freedom to choose between manufacturers. Multimedia communication is greatly dependant on good standards, and hence standards are a prerequisite for effective communication.

### 2.3.1 Still Image Compression

#### 2.3.1.1 JPEG

In the 1980's a joint ISO/CCITT committee was formed to determine standards for still image compression including both lossless and lossy modes. This committee was then named JPEG (Joint Photographic Experts Group) and they have established the first international compression standard for continuous-tone still images, for both greyscale and colour textures.

a) Algorithm

The JPEG standard includes two basic compression methods namely lossless compression, which is based on a predictive method, and the simple lossy technique, which is based on the Baseline method [69]. Depending on the type of image, a selection of the lossless or lossy technique can be made. For example, images that contain text will be coded using the lossless type since the text cannot be sacrificed. However, most of the time, the lossy technique is preferred.

The JPEG compression scheme is based on local approximation. The processing steps are depicted in Figure 2-5.

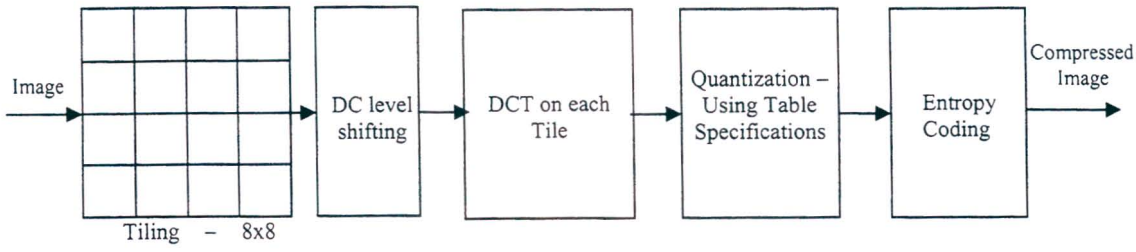



Figure 2-5: Block diagram of the JPEG compression scheme

The source image is initially divided into non-overlapping blocks, normally of size 8x8, and processed individually. Each block is then shifted from unsigned integers to signed integers. This shifting is required as to feed the forward DCT. The DCT has the effect of splitting each sample in the block into a unique 2-D spatial frequency, which comprises the input signals spectrum. The DCT also concentrates most of the signal energy in the lower spatial frequencies. Thus in the resultant transformed matrix, low frequency information appear in the upper left hand corner of the matrix while that of the higher frequencies in the lower right hand corner. A quantization method then needs to be applied to the transformed coefficients. Each coefficient is uniformly quantized according to a quantization table which must be specified by the application (or user). In other words, each coefficient in the transform is divided by its corresponding entry in the quantization table, and rounded off to the nearest integer. A set of quantization tables appear in the standard. Higher frequencies are normally discarded, since the human eye is insensitive to these frequencies. Normally, JPEG uses a luminance matrix as its quantizer, where each entry is based on a visual threshold of its corresponding basis elements. Thus the quantizer matrix table contains larger entries at the lower right hand corner and small entries at the upper left hand corner. This is done as to preserve the lower frequency components and highly attenuate the higher frequency ones. The final step of the coder is to encode the output of the quantizer. Before encoding the coefficients, they are arranged in order of frequency (zigzag sequence – Figure 2-2), from low to high. This is done since the low frequency information is more important and needs to be transferred first. The entropy coding stage is performed by either the Huffman or Arithmetic Coder.

b) Significant Features of JPEG

The JPEG standard was developed for continuous tone image compression, which has the following properties [69]:

- Be state of the art in terms of compression rate over a range of image quality ratings. The user can set the desired quality/compression trade-off.
- Applicable to any kind of continuous tone digital image.
- Have tractable computational complexity.

- Sequential, Progressive and Hierarchical encoding – Allows the user to build up the image to the required resolution i.e. the image is encoded at multiple resolutions so that lower resolution versions may be accessed without decompressing the image at its full resolution. The progressive mode encodes the quantized coefficients by a mixture of spectral selection and successive approximation, while hierarchical mode uses a pyramidal approach to computing the DCT coefficients in a multi-resolution method.

#### 2.3.1.2 JPEG-2000

With the increasing rise of multimedia and internet applications, the JPEG standard was not sufficient to produce successful results. The current JPEG was optimized for natural images, and depicted poor performance for images containing a mixtures of text, bi-level and natural objects. Furthermore, JPEG could not provide low bit-rate compression, since blocking artefacts would appear in the decompressed image. Thus there was a need to create a new image compression system that could cater for different types of still images (grey level, colour, bi-level) that contain different characteristics (such as text, natural images, rendered graphics, etc). Also, this coding system should provide a low bit-rate operation with a quality performance superior to existing standards and a need for real time transmission. The JPEG-2000 standard was not meant to replace JPEG, but merely to complement its standards.

a) Algorithm

The JPEG-2000 standard also uses the transform coder paradigm. Research has shown that the DWT offers more flexibility in terms of image compression than the DCT, especially at higher compression ratios. Hence the JPEG-2000 has adopted a wavelet based solution. Furthermore, the DWT can provide an integer wavelet transform and hence a lossless compression system can be developed, as oppose to DCT, where transform coefficients are floating point, which results in rounding errors.

The outline of the JPEG-2000 compression algorithm is depicted in Figure 2-6.

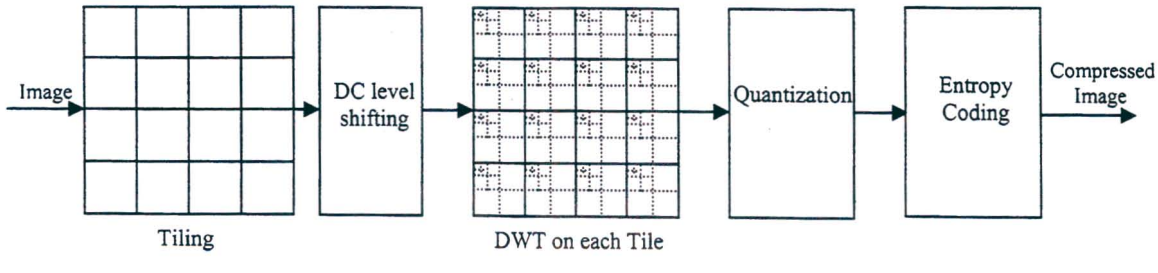



Figure 2-6: Encoding structure of the JPEG-2000 standard

This algorithm operates on tiles of the image where initially the image is divided into non-overlapping blocks (or tiles). Each image tile is processed individually. Prior to computation of the forward DWT on each tile, all samples of the image tile are DC level shifted by subtracting the same quantity from each sample. A type of a scalar quantizer is used in the standard, in which the coefficients are reduced in precision. Each block tile is then entropy coded using the arithmetic encoder. At the quantization stage bit-plane coding is employed i.e. coding the most significant bits first. The bit-plane coding normally contains three passes, and each pass is coded independently. This allows the decoder to decode the image at a given image quality. Hence the system has the ability to decode from a single stream, different spatial resolutions of the image.

b) Features of the JPEG-2000 Standard

The following features are contained in the standard [70]:

- Superior low bit-rate performance.
- Compression of continuous-tone and bi-level images – Able to compress images with text, rendered graphics, natural images, etc. Also can compress images with various dynamic ranges, such as 1 bit to 16 bit images.
- Able to achieve lossless and lossy compression.
- Progressive transmission by pixel accuracy and resolution – Can achieve reconstructions at different resolutions. This occurs since the DWT splits the images into a number of subbands and lower resolutions can be achieved by decoding the appropriate number of subbands.
- Random code stream access and processing – Allows the user to define important parts of the image (regions of interest (ROI)), that will be randomly accessed and decompressed with less distortion than the rest of the image.
- Robustness to bit-errors – Error correction algorithms are included to ensure that significant potions of the code stream are not affected by errors.
- Open architecture – Allowance of the system to optimize compression for different image types and applications.
- Real time coding and decoding.

### 2.3.2 Video Compression

Video compression standards have been introduced as a result of different multimedia applications. The major difference between these standards lies in the bit-rates that they produce for applications they are targeted for. Table 2-1 details the different video compression standards, their operating bit-rates as well as applications they are developed for.

| Video Compression Standard | Market | Video Bit-rate | Frame accurate editing | Scalability | Still Image Mode | Lossless Mode |
|---|---|---|---|---|---|---|
| MPEG-1 | Video CD authoring | 1.0-1.5 Mbit/s @ 352x240, 29.97 fps | No | Low | No | No |
| MPEG-2 | DVD authoring | 3.0-100.0 Mbit/s @ 720x480, 29.97 fps | No | Low | No | No |
| MPEG-4 | Internet Streaming | 0.3-1.0 Mbit/s @ 352x240, 29.97 fps | No | High | Yes | No |
| MJPEG | Video Production | 10.0-80.0 Mbit/s @ 720x480, 29.97 fps | Yes | Low | Yes | No |
| DV | Professional Video Production. Digital Video Cameras. | 25.0 Mbit/s @ 720x480, 29.97 fps | Yes | Low | No | No |
| MJPEG2000 | Professional Video Production. Digital Video Cameras. Video/Image streaming. | 2.0-50.0 Mbit/s @ 720x480, 29.97 fps | Yes | High | Yes | Yes |
| ITU-T H.263 | Video Phone | < 0.1 Mbit/s | No | High | No | No |

Table 2-1: Video compression standards for different applications [71]

Since we are concentrating on multimedia applications that require low bit-rate compression, the MPEG-4 and H.263 compression standards will be discussed.

#### 2.3.2.1 H.263 +

The H.263 standards are the latest standard for low-bit rate video compression developed by ITU-T. The H.263+, which is the second version of the H.263 standard, was developed by volunteers in open committees. These standards combine features of the MPEG and H.261 standards for very low bit rate coding. H.263+ supports picture formats in Table 1-1 (such as QCIF) as well as custom picture formats.

a) Algorithm

The H.263 coding standards uses block based coding, in which each picture frame is sub-divided into smaller blocks (normally 8x8), and is processed one by one. A macro-block (MB) is defined as a collection of four blocks in the luminance frame (each of which is of size 8x8) and the corresponding blocks from the chrominance frames, Cr and Cb. A number of macro-blocks are grouped together to form group of blocks (GOB). The H.263 allows GOB to contain one or more rows of macro-blocks. Each macro-block can be coded as intra or inter. In the H.263 algorithm, spatial redundancy is removed by the DCT (intra frame coding), while motion estimation and compensation are used to remove temporal redundancies (inter-frame coding).

Intra-frame coding takes the approach of the JPEG image compression standard, in that the transformed coefficients are quantized according to the zigzag scan, to produce a one dimensional array. This array is then coded using run-length coding, with the Huffman coder being the entropy coder. H.263 supports block based motion estimation and compensation, which is performed at the MB level [67]. The compensation process involves finding the best matching MB in the previous frame for every MB in the current frame. The offset between the MB in the current frame and the corresponding matching MB in the previous frame is stored. This offset is called the motion vector. Once all predictions for the MBs are found, the prediction frame is constructed. Thereafter, the difference (or residue frame) is calculated, i.e. the difference between the prediction frame and the previous frame, and is coded using intra-frame coding technique. The difference frame will contain information that has not been coded by motion estimation.

b) Advanced Options

Several options are supported to improve the performance of the H.263 compression standard. These advanced options are provided for better coding efficiency and error resilience. The

options are listed below, with the first four being supported by H.263 and the rest are enhancements that have been made for the H.263+ standard [67].

- Unrestricted motion vector mode – This allows motion vectors to point outside a picture boundary.
- Syntax based arithmetic coding – The arithmetic coder is used instead of the Huffman coding scheme.
- Advanced prediction mode – Uses overlapped block motion compensation and also uses four motion vectors per MB.
- PB Frame mode – Allows a P frame and a B-frame to be coded together as one PB frame. A P-picture is predicted from the previous frame, while the B-picture is predicted from the following or previous frames.
- Advanced intra coding – Improves the compression efficiency while coding intra MBs in a given frame, using spatial prediction of DCT coefficient values.
- De-blocking filter mode – An adaptive filter is applied to each block edge to reduce the blocking artefacts.
- Slice structured mode – Helps to improve error resilience, by grouping a number of MBs.
- Reference picture selection mode – Allows a selection of any previously decoded frames which is used as the reference frame to predict the current frame.
- Scalability related enhancements – allows for decoding of the video sequence at different quality levels.

#### 2.3.2.2 MPEG-4

MPEG (Moving Picture Experts Group) is an ISO/IEC working group developing international standards for video compression and decompression. The MPEG-4 standard was designed to be the global multimedia language, targeted for applications that require low bit-rates. MPEG-4 provides numerous video coding schemes that present a generic tool for the transmission and storage for various required applications. MPEG-4 uses media objects to represent visual or audio content. Media objects can be of synthetic or natural origin. The MPEG-4 image and video coding algorithms gives an efficient representation of these visual objects of arbitrary shape, with the goal to support content-based functionalities. Furthermore, the standard allows the user to interact with the objects in the scene, within the limits set by the author [74].

a) Algorithm

The algorithm is similar to that of the H.263 standard. The main steps of the coding scheme are:

- Division of the picture into MB (of size 16x16).

- Motion estimation and compensation – Frame differencing.
- Transform coding with DCT.
- Quantization.
- Run-length and Huffman coding.

The coding structure also involves shape coding for arbitrarily shaped visual objects. The compression efficiency can be significantly improved by using dedicated object-based motion prediction tools for each object in a scene. A number of motion prediction techniques have been defined in the standard. These are:

- Standard 8x8 or 16x16 pixel block-based ME/MC.
- Global MC for video objects - using 8 motion parameters that describe an affine transformation.
- Quarter-Pel Motion Compensation - The spatial resolution of the motion compensation is increased to 1/4 pixel.
- Shape-adaptive DCT - In the area of texture coding, the shape-adaptive DCT improves the coding efficiency of arbitrary shaped objects.

b) Features of the MPEG-4 Standard

The MPEG-4 standard provides solutions and algorithms for [74]:

- Efficient compression of images and video.
- Efficient compression of textures for texture mapping on 2-D and 3-D meshes.
- Efficient compression of implicit 2-D meshes.
- Efficient compression of time-varying geometry streams that animate meshes.
- Efficient random access to all types of visual objects.
- Extended manipulation functionality for images and video sequences.
- Content-based coding of images and video.
- Content-based scalability of textures, images and video.
- Spatial, temporal and quality scalability – scalability allows decoding a part of the coded stream to construct images with reduced decoder complexity, reduced spatial resolution, reduced temporal resolution, or reduced video quality.
- Error robustness and resilience in error prone environments – three tools are provided, namely resynchronization tools, data recovery tools and error concealment tools.
- Support for Video format (PAL, NTSC), Pixel aspect ratio (e.g. 1:1, 12:11 for 4:3 PAL, 10:11 for 4:3 NTSC), Source frame rate (e.g. 25 fps, 29.97 fps, 30 fps).

## 2.4 Summary

The transform coder utilizes the statistical properties of the mathematical transformation used in its system. All compression occurs at the quantizing and entropy coding stages. The quantizing process, which is lossy, reduces the precision of the transformed coefficients. On the other hand the entropy coding stage is lossless, and two algorithms, namely the Huffman and Arithmetic coders, were discussed.

The compression ratio and the bit-rate are used to define the amount of compression an algorithm can achieve. Furthermore, the PSNR is a commonly used measure of the recovered image quality. These measurements are used to compare the performances of different compression systems.

This section has also presented the major compression standards that exist for image and video coding. The image compression standards utilize the transform coding concept, in their algorithms. The JPEG standard uses the DCT at the transform stage, whereas the JPEG-2000 uses the DWT. In all video compression standards, temporal decorrelation is achieved by a ME/MC approach. These standards will provide a useful comparison for the different fractal and wavelet coding techniques developed in the following chapters.

# CHAPTER 3 - FRACTALS AND ITERATED FUNCTION SYSTEMS

The word fractal originated from research of Mandelbroth, which was used to describe objects that are too irregular to be described by traditional geometry. Fractal is said to be derived from the Latin word "*fractus*", meaning broken [13]. Mandelbroth noticed that any regular object is related to scale (see Figure 3-1). In Figure 3-1(a), observing a circle over a large rectangular window, you will see a circle. When the observation window is reduced (Figure 3-1(b)), you will then regard the circle as an arc. Further reduction of the observation window, will cause you to perceive the circle as a number of tiny lines (Figure 3-1(c)). However, Mandelbroth showed that with many natural and man-made phenomena, as the observation window reduces or expands, the complexity and shape of these phenomena remains unchanged [13]. He then termed these phenomena, which show characteristics of invariance under scale changes, as fractals.

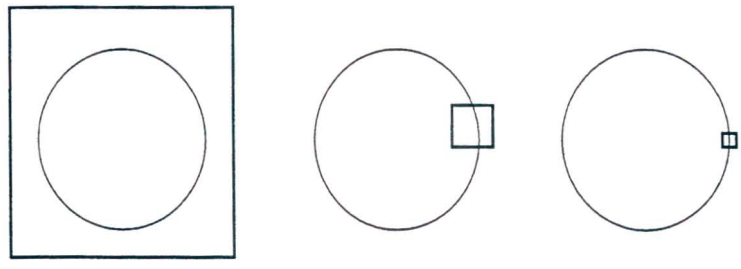



Figure 3-1: Observing a circle at different scales [13]

Michael Barnsley [1] described a fractal as a geometric form whose irregular details reappear at different scales and angles. He then determined that these details can be described by a set of affine or fractal transforms. On the other hand, Falconer proposed that it is best to regard a fractal as a set that has certain properties, rather than looking for a precise definition [10]. The various properties of a fractal set are as follows:

- It is too irregular to be described in traditional geometrical language, both locally and globally.
- It has a fine structure and detail in every scale.
- It usually has some form of self-similarity, either approximate or statistical.
- In most cases, it can be described by a simple algorithm. This algorithm may be iterative by use of affine transformations.

The self-similarity is the most important characteristic of fractals, and distinguishes a fractal set from the regular set.

Section 3.1 provides a brief introduction to fractal theory and iterated function systems. Section 3.2 uses the theorems presented in the previous section to develop an image coder, called the fractal block coder. The details of all the algorithms used in the compression scheme are presented. The next section (Section 3.3) extends the fractal image compression idea to code video sequences.

## 3.1 Theory of Fractals and the Fractal Transform

Fractal theory is based upon transformations that can generate self-similar sets. Michael Barnsley termed these transformations as Iterated Function Systems (IFS). The fundamental principle of fractal theory or IFS theory is the contraction mapping principle. This principle guarantees the convergence of the IFS to a unique output, no matter what the initial input is. Barnsley also observed that very simple IFS can generate complex sets with infinite detail that represent natural objects. He then posed the question, "if IFS can generate natural objects, can the inverse be done?" This question implies generating IFS from natural images. The collage theorem helps us to understand this problem. The theory of IFS is relatively straight forward; therefore the most important principles, theorems, definitions and proofs will be presented. However, certain theorems will just be cited without proofs. A full introduction to fractal theory and IFS is presented in [1].

### 3.1.1 Iteration

Iteration is a process, or set of rules, which one repeatedly applies to an initial state. The main aim of iteration is to reach a desired result or a final state after repeating the task on an initial value or state.

An example of an iteration is to apply the square root function to an initial value (or seed) $x_0$ and then take the square root of the output. This process is then repeated $n$ times to achieve the final result [7]. This iteration procedure is shown below, with $F(x)=\sqrt{x}$, $x \in \Re$, $n=0,1,2,...$ and $F^m(x_0)$ denoting the $m^{th}$ iteration.

$$x_1 = F^{(1)}(x_0) = F(x_0) = \sqrt{x_0} = x_0^{\frac{1}{2}}$$

$$x_2 = F^{(2)}(x_0) = F(x_1) = \sqrt{x_1} = \left(x_0^{\frac{1}{2}}\right)^{\frac{1}{2}} = x_0^{\frac{1}{4}}$$

$$x_3 = F^{(3)}(x_0) = F(x_2) = \sqrt{x_2} = \left(x_0^{\frac{1}{4}}\right)^{\frac{1}{2}} = x_0^{\frac{1}{8}}$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$x_n = F^{(n)}(x_0) = F(x_{n-1}) = \sqrt{x_{n-1}} = x_0^{\frac{1}{2^n}}$$

Figure 3-2: Iteration process for the square root function

The iteration rule, $x_n = x_0^{1/2^n}$ produces a set of outputs (for $n = 0,1,2,\ldots$), with each depending on the initial value $x_0$. Thus for different initial values, the behaviour of the iterative function could be different. In the above case, if $x_0 = 1$, the output of the function will always be 1 and will not change. Therefore it can be pointed out that the function has a fixed point for $x_0 = 1$. If $x_0 > 1$, the above function will reach 1, the same fixed point, after a larger number of iterations. Also, if $0 < x_0 < 1$, the function will approach 1. On the other hand, if $x_0 = 0$, the output of the function will always be 0, another fixed point. Thus the above iterative function has two stable fixed points, namely 1 and 0. Depending on the value of $x_0$, the function will converge to either of the above fixed point. Thus this iterative function is termed convergent or contractive.

There also exist iterative functions that are divergent, i.e. the output approaches infinity. One such function is the square function, where $F(x) = x^2$ and $x > 0$. The iterative process of this function is shown below.

$$x_1 = F(x_0) = x_0^{\,2}$$

$$x_2 = F(x_1) = x_1^{\,2} = \left(x_0^{\,2}\right)^2 = x_0^{\,4}$$

$$x_3 = F(x_2) = x_2^{\,2} = \left(x_0^{\,4}\right)^2 = x_0^{\,8}$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$x_n = F(x_{n-1}) = x_{n-1}^{\,2} = x_0^{\,2^n}$$

Figure 3-3: Iterative process of the square function

The above square function is divergent if $x_0 > 1$. However, if $x_0 = 1$, the function will converge to 1 and if $0 < x_0 < 1$, the output will converge 0. Therefore, the square function is both

convergent and divergent and this depends on the initial value. There also exist iterative functions that are periodic and divergent only.

### 3.1.2 Copying Machine Algorithm

The copying machine algorithm (introduced by Kominek [2]) is an elegant method to introduce fractals and iterated function systems. Consider a copying machine that is able to make three copies of the source at once. Each copy is reduced in size by a factor of two, and then translated, as shown in Figure 3-4. Lets now consider what will happen if we feed the output back into the copying machine, with the same set of rules. The resulting outputs are also depicted in Figure 3-4 for a specific number of iterations. The final figure produced (after ten iterations) is the well known iterated function system, the Sierpinski Triangle, which is a fractal.

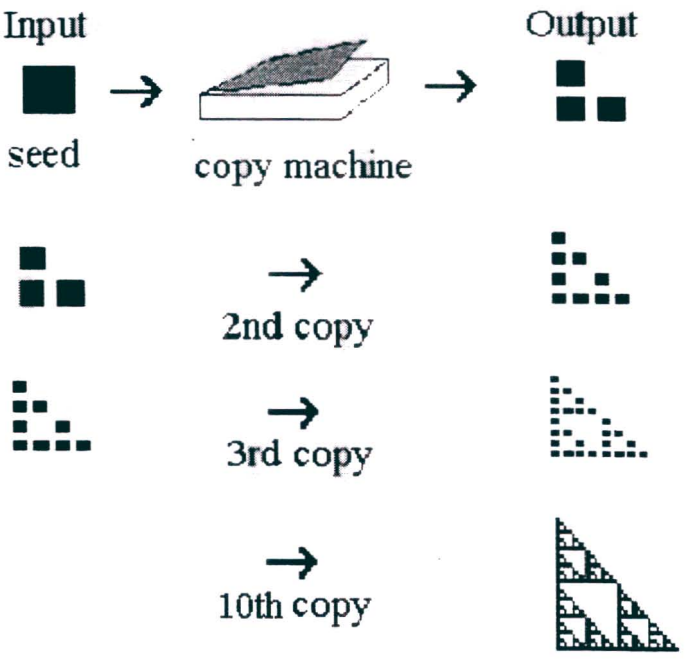
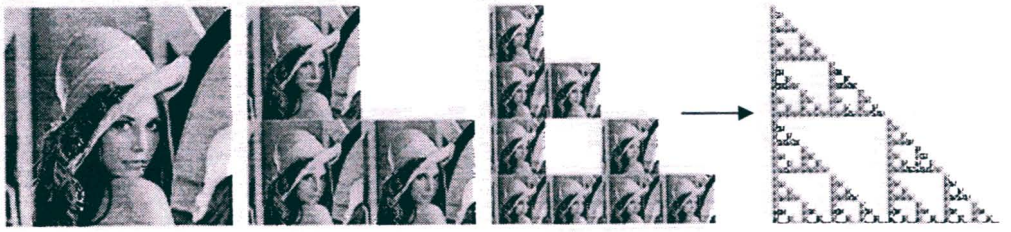



Figure 3-4: Copying machine using three rules to produce the Sierpinski triangle [7]

Suppose that the initial input to the copying machine is changed from a square to the "Lena" image (Figure 3-5). After the same number of iterations, the Sierpinski Triangle is still produced. Therefore, it does not matter what picture the user begins with, as the copying machine will still produce the Sierpinski Triangle. Thus it is the set of rules of the copying machine that determines the final output of the iterated function system. The final output is called the attractor.



Original 1st Iteration 2nd Iteration 10th Iteration

Figure 3-5: Output of the copying machine with the "Lena" image as the input

Since the copying machine produces three copies, there must be three set of rules that when combined produces the attractor. The rules are in the form of affine transformations as shown below:

$$
\begin{aligned}
w_1\begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \\
w_2\begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \\
w_3\begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}
\end{aligned}
\tag{3.1}
$$

Each transformation scales the input, with $w_2$ and $w_3$ translating the scaled input in the vertical and horizontal direction respectively.

Let us now compare the Sierpinski triangle to the properties of fractals. The self similarity property is evident, in that the triangle contains itself over and over again. From Figure 3-4 and Figure 3-5, if we zoom into the triangle at any stage, we will still encounter the triangle and never reach a stage where we would see the initial seed. Thus the Sierpinski triangle has detail in every scale. The algorithm to generate the Sierpinski triangle consists of three simple transformations (see (3.1)). Thus fractals can be generated by mathematical transformations. However, these transformations must be constrained with the limitation that they must be contractive. This means that a given transformation applied to any two points in the input image must bring them closer at the output.

### 3.1.3 Metric Spaces and Mappings

As alluded to earlier, to produce an IFS that converges to a final image (or attractor), the IFS must be a contractive mapping. This means that given a pair of points, and if the IFS is applied on these points, the resultant output after each iteration must bring the points closer together. Therefore a metric space together with a distance function is required which is able to measure

the distance between image points. This metric space will then determine which function mappings are contractive or not.

**Definition 3-1: Metric Space**: A metric space $(X,d)$ is a set $X$ together with a real valued function $d$, which measures the distance between pairs of points $x$ and $y$ in $X$. $d$ is called a metric on the space $X$ when it has the following properties [7]:

$$\begin{aligned} &i)\ \ d(x,y)=d(y,x),\forall x,y\in X \\ &ii)\ \ d(x,y)\geq 0,\forall x,y\in X \\ &iii)\ \ d(x,y)=0 \text{ if and only if } x=y,\forall x,y\in X \\ &iv)\ \ d(x,y)\leq d(x,z)+d(z,y),\forall x,y,z\in X \end{aligned} \tag{3.2}$$

If the metric space is the Euclidean two dimensional space, $\mathbb{R}^2$, the Euclidean distance metric is given by:

$$d(x,y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2},\forall x,y\in\mathbb{R}^2, \tag{3.3}$$

where $x=(x_1,x_2)$ and $y=(y_1,y_2)$.

**Definition 3-2: Transformation or Mapping**: A transformation or mapping on a space $X$ is defined as a function $f:X\to X$ [7]. The function $f$ is one to one, i.e. given $x,y\in X$, then if $f(x)=f(y)$, this implies that $x=y$. The function $f$ is also invertible, i.e. if $f(x)=y$, there exists a function $f^{-1}:X\to X$, such that $f^{-1}(y)=x$, the unique point. $f^{-1}$ is called the inverse of $f$.

### 3.1.4 Convergence and the Contraction Mapping Principle

**Definition 3-3: Attractor**: Consider a function $f:X\to X$. If there exists a $y\in X$ such that

$$\lim_{n\to\infty} f^{(n)}(x)=y,\forall x,y\in X, \tag{3.4}$$

then $y$ is called the attractor of $f$ [8].

**Definition 3-4: Fixed Point**: If a function $f:X\to X$ has the property that $f(x)=x,\ x\in X$, then $x$ is called the fixed point of $f$, and is denoted as $x_F$ [8].

In fractal theory, the terms attractor and fixed point are used interchangeably, since all attractors are fixed points by definition of iteration.

**Definition 3-5: Cauchy Sequence**: A sequence $\{x_n\}_{n=1}^{\infty}$ of points in a metric space $(\boldsymbol{X},d)$ is called a Cauchy sequence if for any number $\varepsilon > 0$, there exists an integer $N > 0$ such that $d(x_n, x_m) < \varepsilon$ for all $n, m > N$ [7].

One can think of the sequence $\{x_n\}_{n=1}^{\infty}$ being generated iteratively by a function $f$ after $n$ iterations. Thus as $m$ and $n$ grow relatively large (or after a large number of iterations), the difference in values of the sequence becomes smaller.

**Definition 3-6: Converging Sequence**: A sequence $\{x_n\}_{n=1}^{\infty}$ of points in a metric space $(\boldsymbol{X},d)$ is said to converge to a point $x \in \boldsymbol{X}$, if for any number $\varepsilon > 0$ there exists an integer $N > 0$, such that $d(x_n, x) < \varepsilon$ for all $n > N$ [7].

In a converging sequence, the distance measure is between the points in the sequence and a fixed point to which the sequence is approaching and not between consecutive points, as in the Cauchy sequence.

**Definition 3-7: Complete Metric Space**: A metric space $(\boldsymbol{X},d)$ is complete if every Cauchy sequence $\{x_n\}_{n=1}^{\infty}$ in $\boldsymbol{X}$ has a limit $x \in \boldsymbol{X}$ [7].

**Definition 3-8: Lipschitz constant**: Consider a function $f : \boldsymbol{X} \to \boldsymbol{X}$ on a metric space $(\boldsymbol{X},d)$. If there exists an $s \in [0, \infty]$ such that,

$$d(f(x), f(y)) \leq s\, d(x, y) \quad \forall x, y \in \boldsymbol{X}, \tag{3.5}$$

then $f$ is called the Lipschitz on $\boldsymbol{X}$, and $s$ is called the Lipschitz constant for $f$ [8].

**Definition 3-9: Contraction mapping**: If a function $f : \boldsymbol{X} \to \boldsymbol{X}$ has a Lipschitz constant $0 < s < 1$, then $f$ is called contractive or a contraction mapping. The number $s$ is then called the contractivity factor for $f$.

**Theorem 3-1: Contraction mapping principle (CMP)**: Let $(\boldsymbol{X},d)$ be a complete metric space and $f : \boldsymbol{X} \to \boldsymbol{X}$ a contraction mapping with a contractivity factor $s$ [8]. Then,

(i) $f$ has a unique fixed point $x_F \in \boldsymbol{X}$.

(ii) Moreover, for any point $x \in \boldsymbol{X}$, the sequence $\{f^{(n)}(x) : n = 0,1,2,\ldots\}$ converges to $x_F$, i.e. $x_F$ is an attractor of $f$.

PROOF: Given a sequence $\{x_n\}_{n=1}^{\infty}$ on a metric space $(\boldsymbol{X},d)$, with $x_{n+1} = f(x_n)$. Choose any $x_0 \in \boldsymbol{X}$ [8].

Now for any $n, m \in \mathbb{N}$, with $m > n$,

$$d(x_n, x_m) = d(f^{(n)}x_0, f^{(m)}x_0) \tag{3.6}$$

Since $f$ is contractive, substituting (3.5) in (3.6) we obtain,

$$d(x_n, x_m) \le s\, d(f^{(n-1)}x_0, f^{(m-1)}x_0) \tag{3.7}$$

If $f$ is iterated $n$ times, (3.7) becomes

$$d(x_n, x_m) \le s^n\, d(x_0, f^{(m-n)}x_0) \tag{3.8}$$

Now let $k = m - n$

$$d(x_0, f^{(m-n)}(x_0)) = d(x_0, f^{(k)}(x_0)) . \tag{3.9}$$

Applying the triangular inequality repetitively to (3.9), produces:

$$\begin{aligned} d(x_0, f^{(k)}(x_0)) &\le d(x_0, f(x_0)) + d(f(x_0), f^{(k)}(x_0)) \\ &\le d(x_0, f(x_0)) + d(f(x_0), f^{(2)}(x_0)) + \ldots + d(f^{(k-1)}(x_0), f^{(k)}(x_0)) \\ &\le d(x_0, f(x_0)) + s\, d(x_0, f(x_0)) + \ldots + s^{k-1}\, d(x_0, f(x_0)) \\ &= \sum_{i=0}^{k-1} s^i d(x_0, f(x_0)) \end{aligned} \tag{3.10}$$

By a comparing (3.10) to a geometric series in $s$,

$$d(x_0, f^{(k)}(x_0)) \le \frac{1}{1-s} d(x_0, f(x_0)) . \tag{3.11}$$

Substituting the result, (3.11), in (3.10) yields

$$d(x_n, x_m) \le \frac{s^n}{1-s} d(x_0, f(x_0)) . \tag{3.12}$$

Since $s < 1$, $d(x_n, x_m) \to 0$ as $n \to \infty$.

Thus by equation (3.12), $\{x_n\}$ is a Cauchy sequence. From Definition 3-7, $\{x_n\}$ converges in the complete space $(\boldsymbol{X},d)$, i.e. there exists a $x_F$ such that $x_n \to x_F$. This completes the proof for part (ii) of the theorem.

It remains to be shown that $x_F$ is unique. Suppose that there is another fixed point $y_F \in \boldsymbol{X}$, such that $x_n \to y_F$. Therefore,

$$\begin{aligned} d(x_F, y_F) &= d(f(x_F), f(y_F)) \\ &\leq s\, d(x_F, y_F) \end{aligned} \tag{3.13}$$

However, since $0 < s < 1$, the inequality in (3.13) is satisfied if and only if $d(x_F, y_F) = 0$. Since d is a metric, this implies that $x_F = y_F$, hence $x_F$ is unique. ■

The contraction mapping principle is then fundamental in the production of fractals through IFS.

### 3.1.5 Complete Metric Space for IFS fractals

**Definition 3-10: Compact subsets**: Given a metric space ($\boldsymbol{X}$,$d$). Let $\boldsymbol{S}$ be a subset of $\boldsymbol{X}$ denoted as $\boldsymbol{S} \subset \boldsymbol{X}$. $\boldsymbol{S}$ is compact if every infinite sequence $\{x_n\}_{n=1}^{\infty}$ in $\boldsymbol{S}$ contains a convergent subsequence [7].

**Definition 3-11: Hausdorff space**: Let ($\boldsymbol{X}$,$d$) be a complete metric space. We define $\boldsymbol{H}(\boldsymbol{X})$, the Hausdorff space, as the space whose points are the compact subsets of $\boldsymbol{X}$, other than the empty set [7].

A point in the Hausdorff space can be thought of as a set of points in the metric space which makes up the $n^{\text{th}}$ level of iteration of an IFS applied to an image. The Hausdorff space is convenient to use when considering real world images and IFS.

**Definition 3-12: Hausdorff Distance**: Let $A, B \in \boldsymbol{H}(\boldsymbol{X})$ and ($\boldsymbol{X}$,$d$) a complete metric space. The Hausdorff distance between the points $A$ and $B$ is defined by:

$$h(A,B) = d(A,B) \vee d(B,A), \tag{3.14}$$

where $d(A,B) = \max[d(x,B), \forall x \in A]$ and $x \vee y$ means the maximum of $x$ and $y$. $h$ is called the Hausdorff metric on $\boldsymbol{H}$ [7].

**Theorem 3-2: Complete metric space**: Let ($\boldsymbol{X}$,$d$) be a complete metric space. Then ($\boldsymbol{H}(\boldsymbol{X})$,$h$) is a complete metric space [7]. Furthermore, if $\{A_n\}_{n=1}^{\infty}$ is a Cauchy sequence in ($\boldsymbol{H}(\boldsymbol{X})$,$h$), then

$$A = \lim_{n\to\infty} A_n \in \boldsymbol{H}(\boldsymbol{X}) \tag{3.15}$$

can be characterized as follows:

$$A = \{x \in \boldsymbol{X} : \text{there exists a Cauchy sequence } \{x_n \in A\} \text{ convergent to } x\} \tag{3.16}$$

This theorem allows one to declare the existence of IFS fractals. With this result, a complete metric space suitable for IFS has been created.

### 3.1.6 Iterated Function systems (IFS)

**Definition 3-13: Iterated function systems**: An IFS consists of a complete metric space $(\boldsymbol{X},d)$ and a finite set of contraction mappings $\{w_1, w_2, \ldots, w_N\}$, where $w_n : \boldsymbol{X} \to \boldsymbol{X}$. Each contraction mapping has a respective contractivity factor $s_n$, for $n = 1,2,\ldots,N$. The IFS can be denoted as $\{\boldsymbol{X}; w_n, n = 1,2,\ldots,N\}$, which has a contractivity factor $s = \max[s_1, s_2, \ldots, s_N]$ [7].

It still remains to be shown that the above system is contractive. This can be achieved by developing the CMP for IFS in the Hausdorff space. First we need to introduce a few lemmas.

**Lemma 3-1**: Let $w : \boldsymbol{X} \to \boldsymbol{X}$ be a contraction mapping on the metric space $(\boldsymbol{X},d)$, with a contractivity factor $s$. Then $w : \boldsymbol{H}(\boldsymbol{X}) \to \boldsymbol{H}(\boldsymbol{X})$ defined by

$$w(A) = \{w(x) : x \in A\} \quad \forall A \in \boldsymbol{H}(\boldsymbol{X}) \tag{3.17}$$

is a contraction mapping on $(\boldsymbol{H}(\boldsymbol{X}),h)$ with a contractivity factor $s$ [7].

The above lemma illustrates that functions in the metric space $\boldsymbol{X}$ will remain in the Hausdorff space. Furthermore, if the function is contractive, the Hausdorff space preserves the contractivity factor.

**Lemma 3-2:** Let $(\boldsymbol{X},d)$ be a metric space and $\{w_1, w_2, \ldots, w_N\}$ be a finite set of contraction mappings on $(\boldsymbol{H}(\boldsymbol{X}),h)$, with contractivity factors $\{s_1, s_2, \ldots, s_N\}$. We define a contraction mapping $W : \boldsymbol{H}(\boldsymbol{X}) \to \boldsymbol{H}(\boldsymbol{X})$ as

$$W(A) = \bigcup_{n=1}^{N} w_n(A) \quad \forall A \in \boldsymbol{H}(\boldsymbol{X}). \tag{3.18}$$

The $W$ has a contractivity factor $s = \max[s_1, s_2, \ldots, s_N]$.

Combining Lemma 3-1 and Lemma 3-2, the CMP can be restated in terms of the Hausdorff space for IFS.

**Theorem 3-3: Contraction mapping principle for IFS**: Let $\{\boldsymbol{X}; w_n, n = 1,2,\ldots,N\}$ be an IFS with a contractivity factor $s$. Then the map $W : \boldsymbol{H}(\boldsymbol{X}) \to \boldsymbol{H}(\boldsymbol{X})$, stated in (3.18), is a contraction mapping on the complete metric space $(\boldsymbol{H}(\boldsymbol{X}), h(d))$ with a contractivity factor $s$ [8]. The unique fixed point of $W$, $A_W \in \boldsymbol{H}(\boldsymbol{X})$, has the following properties:

$$A_W = W(A_W) = \bigcup_{n=1}^{N} w_n(A_W) \tag{3.19}$$

$$A_W = \lim_{n\to\infty} W^{(n)}(B) \quad \forall B \in \boldsymbol{H}(\boldsymbol{X}) \tag{3.20}$$

Thus an IFS is a set of maps on a complete metric space $(\boldsymbol{X}, d)$, with a unique attractor in $\boldsymbol{H}(\boldsymbol{X})$.

### 3.1.7 Affine Transformations

**Definition 3-14: Affine Transformation**: An affine transform is a mapping $W : \boldsymbol{X} \to \boldsymbol{X}$ such that

$$W(x) = T_s(x) + T_t, \tag{3.21}$$

where $T_s$ is a linear scaling transform and $T_t$ represents linear translations.

For images, the space $\boldsymbol{X}$ is normally the Euclidean 2-D plane, $\mathbb{R}^2$. Thus the affine transformation for this space is given by

$$W(x,y) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = T_s(x,y) + T_t, \tag{3.22}$$

with $a, b, c, d, e, f$ being real numbers.

There are different types of affine transformations that will be used to produce IFS. These are detailed below, with illustrations depicted in Figure 3-6.

a) Dilation:

$$W_d(\boldsymbol{x}) = \begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \tag{3.23}$$

Depending on the values of $a$ and $d$, the dilation could contract or stretch $\boldsymbol{x}$.

b) Translation:

$$W(\boldsymbol{x}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \tag{3.24}$$

The translation causes a scalar shift in the $x$ or $y$ direction corresponding to the values of $e$ and $f$ respectively.

c) Reflection: The reflection can be about the x-axis and the y-axis. These can be represented as:

$$W_{rx}(\boldsymbol{x}) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \qquad W_{ry}(\boldsymbol{x}) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.25}$$

respectively.

d) Rotation: The rotation can be clockwise or counter-clockwise by 90°, 180° and 270°.

To construct an affine transformation, the abovementioned transforms can be used alone or combined. The affine transformation used to produce the Sierpinski triangle appears in (3.1).

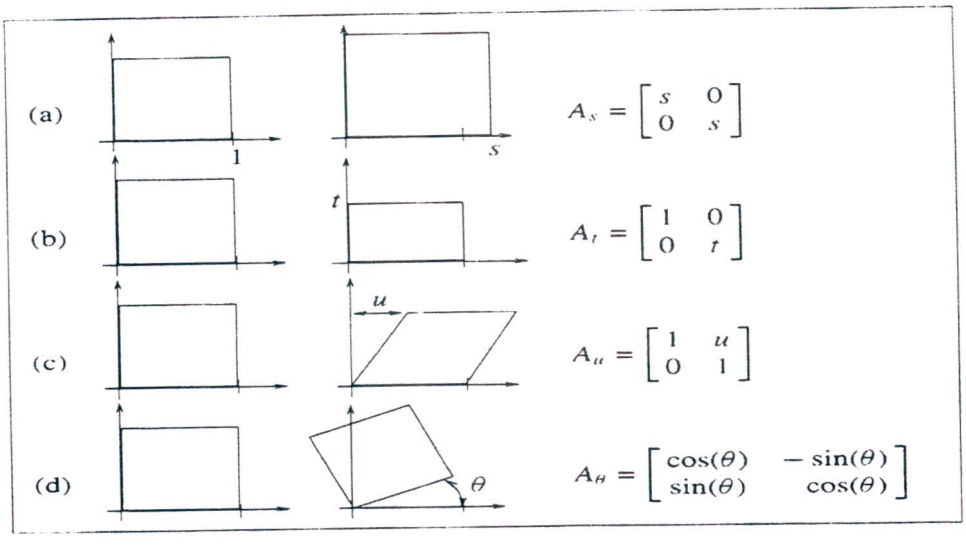



Figure 3-6: Different types of affine transformations [10]
(a) Dilation, (b) Dilation in y direction only, (c) Shearing, (d) Rotation by angle θ

### 3.1.8 The Collage Theorem and the Inverse Problem

We have thus established that fractals generated by IFS allow us to store complex shapes (such as the Sierpinski Triangle) with only a few numbers, i.e. the coefficients of the affine transform. Hence we can store a high resolution image with an insanely high compression ratio. Barnsley was the first who noticed this fact and he posed the question: "If we can generate a fractal of very high complexity with a simple transformation, is it possible to find the transformation to any given image so that the image itself is an attractor?" If this question can be answered in the affirmative, then we could compress any image with this insanely high compression ratio. However, to date, this question cannot be answered fully. But there are proposed solutions (Section 3.2).

**Definition 3-15: Inverse Problem**: The CMP guarantees that a contractive IFS converges to its attractor. Now, given an attractor (such as an image), does a contractive IFS exists or not. This inverse problem can be restated in mathematical terms. Let $(\boldsymbol{X},d)$ be a complete metric space, and let $x \in \boldsymbol{X}$ with $\varepsilon > 0$. Does a contractive $f : \boldsymbol{X} \to \boldsymbol{X}$ exist, with an attractor $x_F$, such that $d(x, x_F) < \varepsilon$?

The collage theorem can help in answering this question. Instead of measuring the distance $d(x, x_F)$, the collage distance $d(x, f(x))$, can be used to simplify the problem.

**Theorem 3-4: Collage theorem**: Let $(\boldsymbol{X},d)$ be a complete metric space. Let $B$ be any given set such that $B \in \boldsymbol{H}(\boldsymbol{X})$, and let $\varepsilon \geq 0$ also be given. Choose an IFS $\{\boldsymbol{X}; w_N, n = 1,2,...,N\}$ with a contractivity factor $0 \leq s < 1$ so that

$$h\left(B, \bigcup_{n=1}^{N} w_n(B)\right) \leq \varepsilon . \tag{3.26}$$

If $A$ is the attractor of the IFS, then from (3.11),

$$h(B, A) \leq \frac{\varepsilon}{1-s} . \tag{3.27}$$

Equivalently,

$$h(B, A) \leq \frac{1}{1-s} h\left(B, \bigcup_{n=1}^{N} w_n(B)\right) \quad \forall B \in \boldsymbol{H}(\boldsymbol{X}) . \tag{3.28}$$

The collage theorem above tells us that by finding maps $w_n$ such that a given set $B$ is mapped close to itself, then $w_n$ will force the attractor of the system to be close to $B$. In other words, to find an IFS whose attractor looks like a given set, we must find a set of contractive transformations on a suitable space, such that the distance between the given set and transformations on that given set is small. That is the union of the transformation of the given set looks like that set.

## 3.2 Fractal Image Compression

In 1984, Pentland was the first to introduce fractal theory in the description of natural scenery, and concluded that the image of a natural scene can be approximated by a fractal set. In 1988, Barnsley and Sloan [13] presented the first fractal image compression scheme. Loosely speaking their method was to find an IFS whose attractor looked close to the original image. As described by Barnsley, "the central goal of fractal image compression is to find resolution

independent models, defined by finite length strings of zeros and ones, for real world images." Hence the coefficients of the IFS were used to store the image, instead of the image itself.

Fractal compression techniques are fundamentally different from the standard transform based compression schemes, such as the DCT. Fractal based techniques exploit the property of self-similarity, in that a part of an image is similar to another part of the same image, under an appropriate transformation. These transformations are represented by IFS. Fractal image compression is the inverse of fractal image generation, in that a set of IFS is sought that represents the image.

The direct method of fractal image compression using IFS is based on the collage theorem (Theorem 3-4). This theorem states that an image can be represented by contractive transforms (IFS), if the fixed point of the contractive transformation is close to the image. Banach's fixed point theorem then guarantees that, within a complete metric space, that the image may be recovered by iterative application of the contractive transformation to an arbitrary initial image of that space [7]. Hence the image is represented by the contractive transform (IFS). Since IFS formulae require a very small amount of data to be represented and stored, very high compression ratios can be achieved with fractal image compression.

### 3.2.1 Fractal Compression Based on IFS Library

In the original fractal compression scheme proposed by Barnsley [14], transformations were sought, that were composed of the union of a number of affine transforms (IFS) on the entire image. This method was unsuccessful, since for natural images, it is very difficult to find IFS whose attractor approximates the whole image. It was then noticed that images consist of a number of small objects, which matched images whose IFS is known. The well known IFS are the fern, leaf, cloud, fence post, Sierpinski triangle and many more (see Figure 3-7).

The new compression system now composed of a segmentation technique that divides the image to be compressed into a number of small objects. Thereafter each segmented portion is compared to all the images whose IFS are known. Thus a library of IFS needs to be generated initially which consists of the IFS parameters and the respective attractor. If the segmented object matches an attractor in the IFS library, then the IFS parameters are stored, yielding a high compression ratio. This technique of compression using the IFS library is very similar to vector quantization, in which each image block is represented with the index of the codeword that is most similar to that particular image block.

The major problem with this method is that no automated encoding algorithm can be found, i.e. there is no process to automatically and accurately segment an image into meaningful objects. Also, the IFS library can be extremely large, hence coding time and a searching method becomes a key problem.

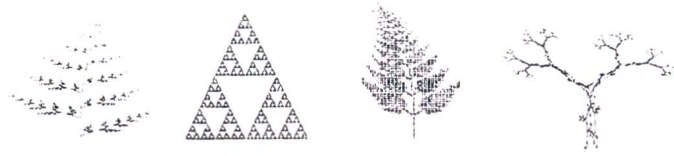


Figure 3-7: Common types of fractals

### 3.2.2 Fractal Compression Based on Local or Partitioned IFS

Fractal compression became a practical reality by the introduction of a compression scheme by A.E. Jacquin in 1991 [5]. He noticed that natural images do not contain the type of self-similarity that can be found in fractals. Hence the failure of the direct and library based compression systems in that the image does not appear to contain affine transformations of themselves. However, natural images contain another form of self-similarity. Jacquin noted that a part of an image is similar to another part of that same image. He then proposed that rather than forming the image from transformations of its whole self, here the image is formed from properly transformed copies of parts of itself. Based on the above observation, Jacquin improved IFS into local IFS, and presented the Fractal Block Coding (FBC) algorithm, which is a practical image coding technique.

#### 3.2.2.1 Local IFS

For all IFS compression methods, the algorithm starts with some target image $T$ which lies in a subset $S \subset \mathbb{R}^2$. This image has to be represented by an IFS. Fractal image compression is achieved by introducing affine transformations, $W_n$, on the set $T$, such that $W_n(T)$ produces a new image with dimensions smaller than that of $T$. This is to ensure a contraction mapping. The coefficients of the transformation $W_n$ must be adjusted such that the new image produced $W_n(T)$ matches a part of $T$. The adjustments made allow the transformation to shrink, rotate, and translate the new image such that a match can be made. Thereafter, a set of transformation $W_1, W_2, \ldots, W_N$ are found such that

$$\tilde{T} = \bigcup_{n=1}^{N} W_n(T), \tag{3.29}$$

where $\tilde{T}$ is the approximation of the target image $T$. The collage theorem guarantees that the attractor, $A$, of the IFS (in (3.29)) will be virtually close to $\tilde{T}$. Therefore if $\tilde{T} = T$, then $A = T$. This implies that image $T$ can be stored by the coefficients of each transformation, $W_n$, hence achieving compression.

The above method of fractal compression, finds IFS for subsets of the image using transformations on the entire image. This is called global IFS. On the other hand, it is possible to find IFS that operate on subsets of the entire image, to produce an image which also is the subset of the entire image. The sub-image can be regenerated by applying the IFS. This is known as local IFS. Hence local IFS act upon domains that are subsets of the entire space, $\boldsymbol{X}$.

**Definition 3-16: Local Contraction Mapping**: Let $(\boldsymbol{X}, d)$ be a compact metric space and let $R$ be a nonempty subset of $\boldsymbol{X}$ i.e. $R \subset \boldsymbol{X}$. Let $w: R \to \boldsymbol{X}$ and let $s$ be a real number such that $0 \le s < 1$. If

$$d(w(x), w(y)) \le s(d(x, y)) \quad \forall x, y \in R, \tag{3.30}$$

then $w$ is called a local contraction mapping on $(\boldsymbol{X}, d)$ and $s$ is the contractivity factor for $w$ [7].

**Definition 3-17: Local IFS**: Let $(\boldsymbol{X}, d)$ be a compact metric space, and let $w_i : R_i \to \boldsymbol{X}$ be a local contraction mapping on $(\boldsymbol{X}, d)$, with a contractivity factor $s_i$, for $i = 1, 2, \ldots, N$ where $N$ is a finite positive integer. Then $\{w_i : R_i \to \boldsymbol{X} : i = 1, 2, \ldots, N\}$ is called a local iterated function system (local IFS). The number $s = \max[s_1, s_2, \ldots, s_N]$ is called the contractivity factor of the local IFS [7]. If $\boldsymbol{S}$ denotes the set of all subsets of $\boldsymbol{X}$, local IFS, $W_{local} : \boldsymbol{S} \to \boldsymbol{S}$, can be redefined as follows:

$$W_{local}(B) = \bigcup_{i=1}^{N} w_i(R_i \cap B) \quad \forall B \in \boldsymbol{S} \tag{3.31}$$

In other words, local IFS compression systems approximate each part of an image by applying a contractive affine transformation on another part of the image.

### 3.2.3 IFS on Grey Maps

Up to this point we have considered IFS to have fixed points that are sets. This means that a pixel is either in the set or not. Hence this is a binary representation and is only suitable for images that are binary, called bitmaps [8]. Typically, we deal with natural images that are either colour or grey level, and not bitmaps. Thus the typical method of applying IFS in the Hausdorff space does not apply anymore. Therefore there is need of some kind of a method to apply IFS

on grey level images. We will be concentrating mainly on greyscale images, since grey images can easily be extended to colour as shown in Section 1.1.1.1.

The above problem was bypassed by the works of Forte and Vrscay [11], who introduced the theory of iterated function systems on grey level maps (IFSM).

**Definition 3-18: IFSM**: Given a complete metric space $(\boldsymbol{X},d)$. An N-map IFS on grey level maps consists of the space $(\boldsymbol{X},d)$ together with two components namely the IFS component and the grey level component [11].

a) IFS Component – $w=\{w_1,w_2,\ldots,w_N\}$, where each $w_n:\boldsymbol{X}\to\boldsymbol{X}$ is a contraction map with a contractivity factor $s_n$.

b) Grey level component – $\phi=\{\phi_1,\phi_2,\ldots,\phi_N\}$, where each $\phi_n:\mathbb{R}\to\mathbb{R}$ is Lipschitz.

The IFSM may be denoted as $\{\boldsymbol{X};w_n,\phi_n,n=1,2,\ldots,N\}$.

The IFSM now performs the fractal transform. It has been shown in [8] that the IFSM is contractive. Since the majority of fractal contractive transforms is performed using affine transforms, it can be extended to incorporate the grey level mapping as follows:

$$W_i\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}a_i & b_i & 0\\ c_i & d_i & 0\\ 0 & 0 & s_i\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}+\begin{pmatrix}e_i\\f_i\\o_i\end{pmatrix}. \tag{3.32}$$

$s_i$ is called the contrast scaling factor and $o_i$ the brightness offset. The IFS component is also known as the geometric transformation and has coefficients (*a, b, c, d, e, f*) in the affine transformations. The grey level component is known as the massic transformation and corresponds to coefficients *s* and *o*. The massic component could make the whole transformation non-contractive, thus it is vital that both transformations be contractive to make the entire affine transform contractive.

### 3.2.4 Jacquin's Method

Jacquin's fractal block coding (FBC) scheme or partitioned iterated function systems (PIFS) is a lossy coding technique. Jacquin's method is based on the principles of local IFS, in which a part of the image is represented by a contractive affine transformation of another part of the same image. The basic idea behind his compression method is to divide the entire image into several small fractals. A brief outline of Jacquin's compression system is provided here, and more details can be found in [5].

The image is divided into non-overlapping range blocks (RB) and domain blocks (DB). The size of the domain blocks is larger than that of the range block. For each of the range block, a domain block is sought, such that the domain block under an appropriate affine transformation matches up to the range block. It must be noted that for an accurate comparison to be performed, the dimensions of the range and transformed domain blocks must be the same. It is also vital that the affine transform performed on the domain block be contractive. This scheme can be spilt into several small steps, shown below (Section 3.2.4.1).

#### 3.2.4.1 Outline of Fractal Block Coders

This section describes the basic steps in the fractal coding algorithm, introduced by Jacquin.

a) Input the Image – Begin the algorithm with a greyscale image of size MxM.

b) Image Segmentation – The entire image is divided into range blocks that do not overlap. For the Jacquin scheme, the blocks are square of size BxB. Hence the segmentation section produces a set of range blocks $R = \{R_1, R_2, ..., R_N\}$. The range blocks are normally coded row-wise.

c) Domain Block Pool – As described earlier, a domain block and a transformation must be found, such that together, they match up to a specific range block. Here, the image is also divided into blocks, but these blocks could overlap. The size of the domain block is normally twice that of the range block, i.e. 2Bx2B. A domain pool is thus formed with the set $D = \{D_1, D_2, ..., D_P\}$. Thereafter a comparison of each range block with every domain block in the pool is performed, one by one, until a best matching domain block is found. The way in which the match is found is discussed below.

d) Affine Transformations – The first step is to perform a geometric transformation of every domain block in the pool. To ensure geometric contractivity, the domain block is reduced in size, so that the dimensions correspond to the range block. To achieve this, a pixel averaging operation is carried out. If the block is to be halved, the pixel averaging operator on a domain block $D_p$ is given by

$$\hat{D}_p(\mu_{x,y}) = \mu_{2x,2y} + \mu_{2x+1,2y} + \mu_{2x,2y+1} + \mu_{2x+1,2y+1}, \qquad (3.33)$$

where $\mu_{x,y}$ is the value of the pixel at coordinate $(x,y)$. Here, each 2x2 pixel block is averaged to produce one pixel. This process is illustrated in Figure 3-8. After the averaging and sub-sampling operation, other geometric transformations, $T_j$, such as a rotations, or reflections can

be performed on the domain block. Once the geometric transformations are completed, the massic transformation needs to be done, as required by the IFSM. To account for the possible darkening of the decompressed image, the average of pixel values of the entire domain block, $\bar{d}_p$ is subtracted from each pixel in that block. A contrast scaling factor, $s_i$, must also be found to make a better match to a range block. A set of scaling factors, $s$, is normally defined by the user, to be a finite set of numbers between 0 and 1. Generally, if $s_i < 1$, then the massic transform is contractive.

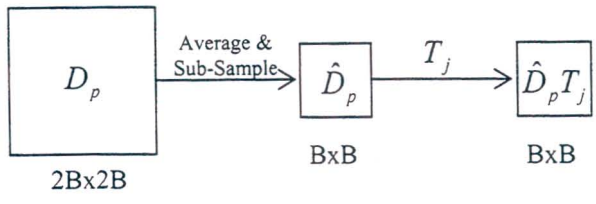



Figure 3-8: Transformations applied to a domain block to create the domain pool

e) Finding a match – For each range block in the set $R = \{R_1, R_2, ..., R_N\}$ a corresponding domain block must be found that under an appropriate contractive transform, matches up to the range block with the minimum amount of error. The error measurement is normally given by the MSE (Equation (2.4)). So for each range block $R_n$, the domain pool is searched. The size of each block in the domain pool $D = \{D_1, D_2, ..., D_P\}$ is initially reduced in and has its mean removed from it. Thereafter every geometric transformation from the set $\{T_1, T_2, ..., T_J\}$ is performed individually on the domain blocks in the pool, and a contrast scaling factor is chosen from the set $s$. A range block, with its mean removed, is then compared to the reduced, transformed and contrast scaled domain block in the search pool via the MSE (see Figure 3-9). The domain block that corresponds to the least amount of MSE is chosen. In mathematical terms the domain block, geometric transformation and contrast scaling factor that minimizes the following equation is chosen,

$$MSE\left[R_n - \bar{r}_n,\ s_i T_j(\hat{D}_p - \bar{d}_p)\right], \tag{3.34}$$

where $\bar{r}_n$ and $\bar{d}_p$ denote the respective means of the range and domain blocks.

f) Representation of the IFS - For each range block $R_n$, the contraction mapping $W_n$ is stored as follows:

- The coordinates of the domain block used, i.e. coordinate of the upper left pixel, $\mu_{x,y}$, is stored.

- Normally, 8 types of geometric transforms are defined, hence the number of the corresponding transform is stored, i.e. 3 bits in this case.
- The number of contrast scaling factors is also pre-defined (normally it is between 4 and 8). Hence the scaling factor is stored using 3 bits.

The encoding algorithm for the FBC is illustrated in Algorithm 3-1. This encoding scheme is time consuming since each range block is compared to all domain blocks in the domain pool. Majority of research is spent finding methods of speeding up the encoding process.

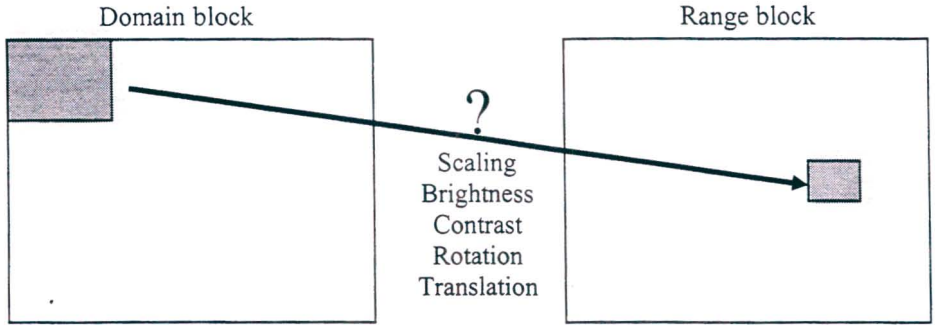



Figure 3-9: Range and domain block matching

**Algorithm 3-1: Fractal Block Coder:**

A $\Leftarrow$ input greyscale image

$\{R_n\} \Leftarrow$ range blocks, with mean $\bar{r}_n$

$\{D_p\} \Leftarrow$ domain blocks, with mean $\bar{d}_p$

**for** each range block $R_n$ **do**

  **for** each domain block $D_p$ **do**

    **for** each transformation $T_j$ **do**

      **for** each scaling factor $s_i$ **do**

        measure error, $En = MSE\left[R_n - \bar{r}_n, s_i T_j(\hat{D}_p - \bar{d}_p)\right]$

        **if** En < Ebest

          {store best mapping values}

          $D_{best} = D_p$, $T_{best} = T_j$, $s_{best} = s_i$, $O_{best} = \bar{r}_n$

        **End if**

      **End for**

    **End for**

  **End for**

  Store $D_{Rn} = D_{best}$, $T_{Rn} = T_{best}$, $s_{Rn} = s_{best}$, $O_{Rn} = O_{best}$

**End for**

### 3.2.4.2 Decoding

When decoding, according to the contractive mapping fixed point theorem (Theorem 3-3), as long as $W_n$ is contractive, application of $W_n$ to an arbitrary image repeatedly will result in a fixed image. The decoding process is fast and simple. Decoding starts with an initial arbitrary image of the same size as the encoded image. The decoder then reads in the list of values from the encoding stream. These values contain the contractive mappings for each range block in a specific order. Since the mappings are contractive, they will converge to an attractor. Seeing that each mapping contains the coordinates of the domain block used, the domain block is extracted from this new image, rescaled and transformed to produce that particular range block. This process is repeated several times for each range block, until the images converges to an attractor. The resulting attractor image is the approximation of the original image. Generally, convergence is achieved after 10 iterations.

**Algorithm 3-2: Decoding of FBC:**

$\bar{A} \Leftarrow$ any initial greyscale image
$\{R_n\} \Leftarrow$ range blocks
$\{D_p\} \Leftarrow$ domain blocks
**for** 1 to number of iterations **do**
  **for** each range block $R_n$ **do**
    $\{D_{Rn}, T_{Rn}, s_{Rn}, O_{Rn}\} \Leftarrow$ read from encoder
    apply map on $D_{Rn} \Rightarrow R_n = [s_{Rn} T_{Rn} (\hat{D}_{Rn} - \bar{d}_{Rn})] + O_{Rn}$
  **end for**
**end for**
$\bar{A}$ now contains the approximation of $A$.

### 3.2.4.3 Performance and Efficiency

The Jacquin algorithm (Algorithm 3-1) was simulated in "Matlab", using fixed block partitioning. The test image used is the "Lena" image which is of size 256 x 256. These simulations were performed on a Pentium-4, 2.4 GHz PC, with 256 MB of RAM. The beauty of fractal image compression is that once the contractive transformations have been defined, we can start with any arbitrary image, apply these transformations, and still recover the original image. Figure 3-10 illustrates this point, where the initial image is a complete shade of grey. One can clearly see that after the first iteration, the majority of the "Lena" image is constructed. Each following iteration brings more clarity to the image. For this test, it took four iterations for the IFS mappings to converge to the attractor, which is the "Lena" image.

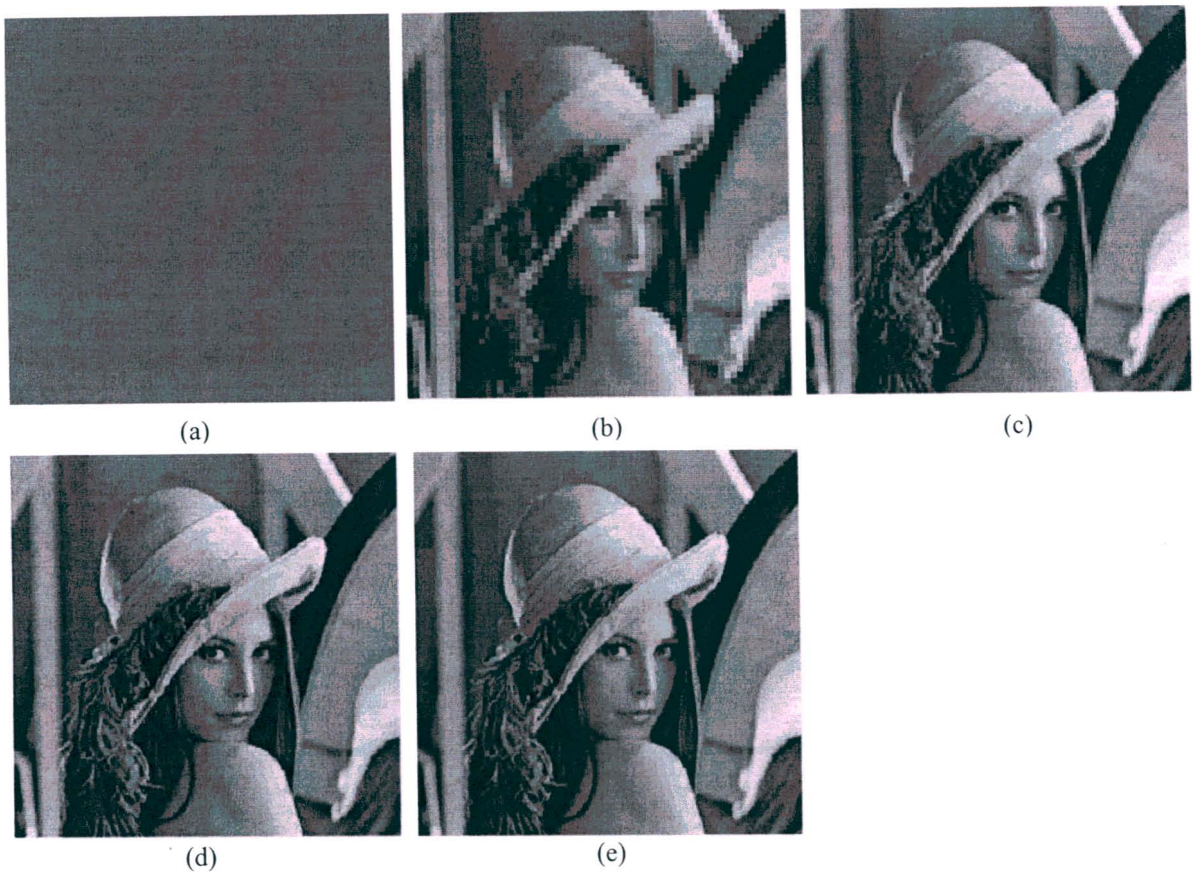



Figure 3-10: Iterative decoding process of a fractal coder
(a) Initial image, (b) 1st iteration - PSNR = 23.45 dB, (c) 2nd iteration - PSNR = 27.24 dB,
(d) 3rd iteration - PSNR = 31.5 dB, (e) 4th iteration - PSNR = 31.5 dB

To assess the compression performance of the fractal block coder, different range block sizes were used. Table 3-1 provides a summary of these results, with Figure 3-11 depicting the recovered images. The most striking feature of fractal compression schemes is the large encoding times. This is a direct result of the block searching method. The greater the number of elements in the domain pool, the larger the computational time will be. We could reduce the size of the domain pool by using less affine transformations, but this will impact on the block matching criteria and reduce the recovered image quality. A solution to this problem is given by the nearest neighbour search (Section 3.2.5.3).

The trend in Table 3-1 is that as the block size increases, the compression ratio increases, and the recovered image quality decreases. This can intuitively be seen as the smaller the range block, the easier it is to find a matching domain block, hence the higher image quality. However, the number of range blocks is greater for smaller block sizes. Thus more transformation parameters need to be stored, accounting for the lower compression ratio. Furthermore, with larger block sizes, the domain search pool reduces, thus reducing the encoding time. It can also be observed, in Table 3-1, that the decoding process is much faster. It

must be pointed out that using other simulation engines, such as "Microsoft Visual C++", will dramatically reduce simulation computational times, since the "Matlab" engine is not optimized for memory management.

Another characteristic of fractal compression systems is that a tiling structure is visualized in the decoded images (see Figure 3-11 (c) and (d)). This structure is formed due to the range block boundaries not being continuous and is more persistent when larger range block sizes are used. A method of removing this tiling structure is to allow range blocks to overlap in the compression stage. However, this will definitely reduce the compression ratio.

| **Range Block Size** | **Compression Ratio** | **PSNR (dB)** | **Encoding Time (s)** | **Decoding Time (s)** |
|---|---|---|---|---|
| 4x4 | 4.41 | 31.50 | 4042 | 12.19 |
| 8x8 | 18.2 | 25.48 | 655.3 | 9.89 |
| 16x16 | 81.9 | 21.34 | 104.6 | 9.17 |
| 32x32 | 356.2 | 17.82 | 22.6 | 9.01 |

Table 3-1: Performance of fractal block coders

For the results in Table 3-1, the MSE (Equation (3.34)) comparison measurement, for range block matching to be successful, was chosen to be 10. Figure 3-12(a) depicts an evaluation of the encoding time of the fractal block coder for different range block matching errors. It can be seen that as the error measurement increases, the quicker it is to find a matching transformed domain block for every range block. This consequently reduces the encoding time. However, Figure 3-12(b) shows that there is a large reduction in the recovered image quality if the range block matching errors are relaxed.


Figure 3-11: Decoded images of the fractal block coder, using different block sizes
(a) Original "Lena" image, (b) 4x4 blocks, (c) 8x8 blocks, (d) 16x16 blocks

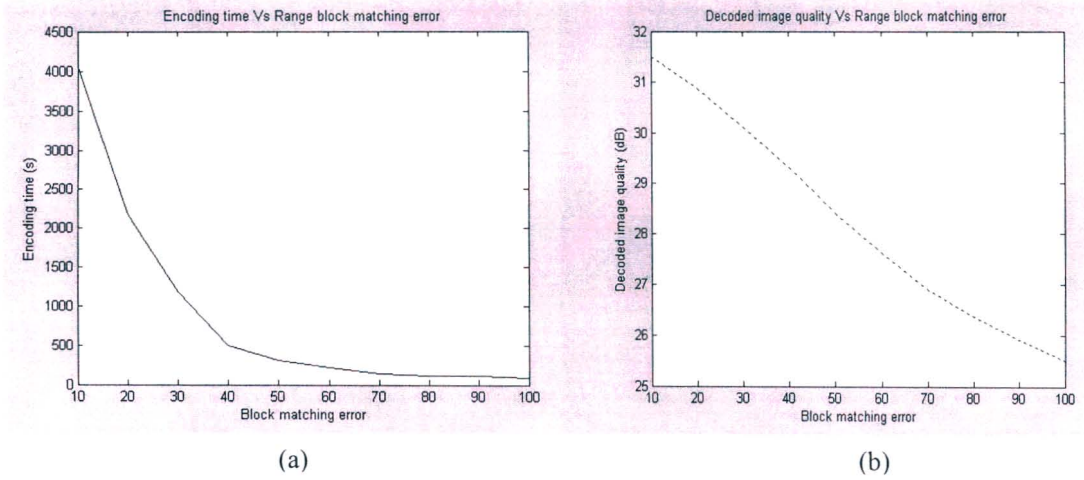



Figure 3-12: (a) Encoding time versus range block matching errors for the fractal block coder
(b) Recovered image quality versus matching errors. The block size chosen was 4x4 on the "Lena" image

### 3.2.5 Factors to Consider

From the above Jacquin compression algorithm, it can be seen that three main issues namely i) how the image is partitioned, and ii) types of contractive transformations, and iii) the type of search used in locating suitable domain blocks are involved in the design and coding of FBC.

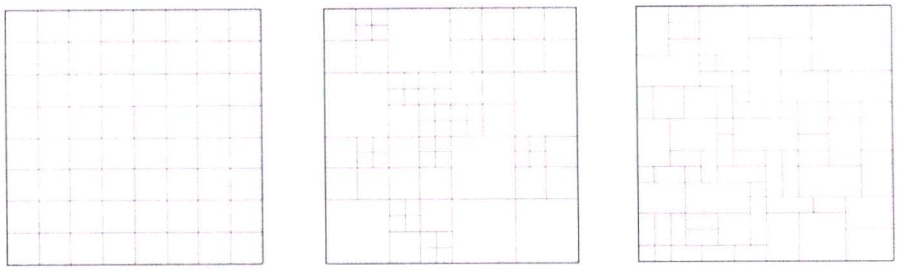
#### 3.2.5.1 Partition Schemes

The first decision to be made when designing fractal coding systems is to choose the type of image partitioning used for range blocks. Since range blocks are the attractors, the types of block transformations become limited, since it depends on the size and shape of the range block. There are a number of partitioning methods, of which the majority being composed of rectangular blocks. The most important point is to partition the image into regions that show self-similarity.

a) Fixed Square Blocks (Jacquin) –This is the simplest partitioning scheme in which the image is divided into non-overlapping square range blocks of a fixed size (depicted in Figure 3-13(a)). It can intuitively be seen that small block sizes will yield small compression ratios and large block sizes, larger compression ratios. However, the clarity of the recovered image suffers with large block sizes.

b) Quadtree [6] – The downfall of the fixed size partition is that the image is partitioned without considering the contents of the image. There are regions of the image that will be covered well using small range block sizes and similarly, there are regions that could be covered well with larger size range blocks. This will increase the compression ratio (larger range block) and maintain the clarity. This observation led to the use of the quadtree partitioning technique. In a quadtree partition, largest range blocks are initially used. If the matching criterion is not satisfied, the size of the range block is halved. This process continues until a matching tolerance, or a minimum block size is reached (see Figure 3-13(b)). It should be clear, that by using this adaptive partition scheme the smallest range blocks will be located in the regions with the highest detail in the image.

c) Horizontal Vertical [6] – The image is segmented into rectangles. If there is no acceptable matching domain block for a given range block, the range block is split into two rectangles either horizontally or vertically (Figure 3-13(c)).



(a) (b) (c)

Figure 3-13: Range block partitioning schemes [6]
(a) Fixed block (b) Quadtree (c) Horizontal-vertical

### 3.2.5.2 Block Transformations

The type of block transform or isometry selected determines the convergence properties in decoding.

a) Spatial Contraction – The spatial contraction of domain blocks is almost universally applied, despite being inessential for the contractivity of the image map as a whole. While contraction by a factor of two in width and height is standard, smaller factors have also been considered. Increasing the spatial contraction to a factor of three has been found to improve decoder convergence [9]. Contraction is usually achieved by the averaging of neighbouring pixels (Equation (3.33)).

b) Rotation and flip operations – Commonly used operations that shuffle pixels in a block.

- identity (no rotation or flip operation)
- orthogonal reflection about mid-vertical axis of block
- orthogonal reflection about mid-horizontal axis of block
- orthogonal reflection about first diagonal of block
- orthogonal reflection about second diagonal of block
- rotation around centre of block, through +90°
- rotation around centre of block, through +180°
- rotation around centre of block, through -90°

c) Contrast and brightness – A contrast scaling factor $s_i$ must be found to make the contrast among pixels in the domain to match the contrast among pixels in the range block. The brightness shift $O_i$ (DC component) must be found to make the brightness of the domain and the range blocks the same. $O_i$ is defined as the difference between the average pixel value in

the range block and average pixel value in the domain block. It is required that block transforms be contractive. It is noted in [9], that the transform can eventually be contractive with respect to the MSE metric if $|s_i| < 1$ for every transformation. The subtraction of the DC component of the domain block prior to scaling creates transformed domains which are orthogonal to the fixed block with the desirable effect of decorrelating the $s_i$ and $O_i$ coefficients.

#### 3.2.5.3 Domain Pool Selection

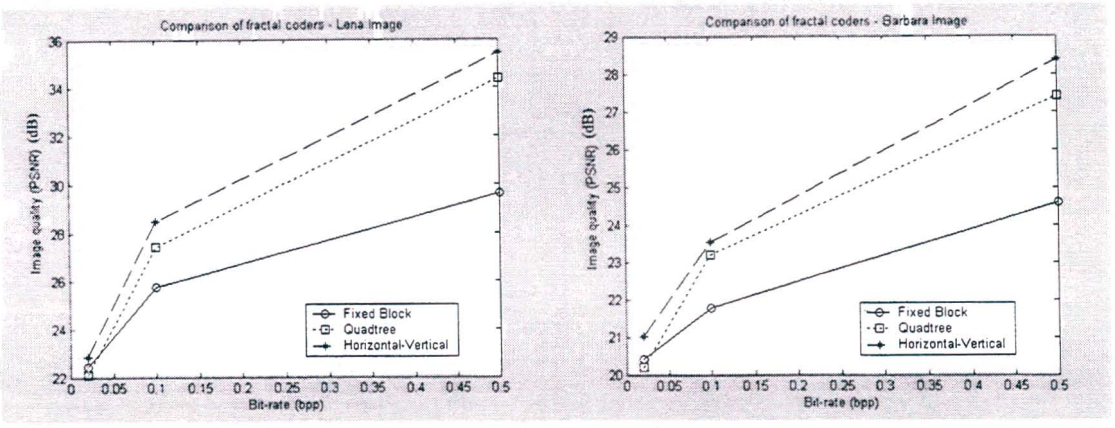
It should be clear that an efficient domain pool creation is crucial, since an increased pool will allow searching over a larger set of domains. Hence, there will be a corresponding increase in the number of bits required to specify the selected domain.

a) Local domain pool – Nearest neighbour search [15]

A number of researchers have noticed a tendency for a range block to be spatially close to the matching domain block. This is based on the observed tendency for distributions of spatial distances between range and matching domain blocks to be highly peaked at zero [9]. Motivated by this observation, the domain pool for each range block may be restricted to a region about the range block [15]. This is known as the nearest neighbour search.

#### 3.2.5.4 Comparison

The performance of the fractal coders using the different partitioning schemes (detailed in Section 3.2.5.1), is revealed in Figure 3-14. This comparison shows that the horizontal-vertical partitioning provides the best rate distortion results. A comparison between the fixed size and quadtree partitioning schemes, show that the fixed size performs better at higher compression ratios (low image quality). This results from the overhead information of range block size and position required by the quadtree method. However, when the required image quality is high, the quadtree method provides better compression and quality. Due to the effectiveness of the horizontal-vertical scheme, advancements have been made to this algorithm. But results show that there is not much gain in the recovered image clarity for a given bit-rate [80]. Another interesting observation of fractal coding is that the larger the image size, the higher the achievable compression ratio. This is because larger images normally contain more spatial redundancy.




(a) (b)

Figure 3-14: Performance comparison of fractal coders using different partitioning schemes (a) "Lena" 512x512, (b) "Barbara" 512x512

## 3.3 Fractal Video Coding

The development of fractal video compression of sequences can be split into two categories namely, inter/intra frame coding and 3-D block coding. Beaumont [17] was the first to investigate the possibility of fractal video coding using the above two approaches. He first tried to extend the Jacquin 2-D fractal image compression technique to 3-D, i.e. he used the fractal technique on 3 dimensional cubes. He found that a high compression could be achieved using this method, but the recovered image sequence quality was poor. Thereafter, he used Jacquin's method to compress each frame in the sequence, by constructing domain blocks from the previous frame (inter-frame). The first frame was compressed using domain blocks from the same frame (intra-frame). It was reported that a 352x288 greyscale sequence could be coded at 80 kbit/s producing decoded frames at a "reasonable quality".

### 3.3.1 Intra/Inter Frame Coding

#### 3.3.1.1 Hurd, Gustava, and Barnsley

In 1992, Hurd, et al [16] developed a fractal compression scheme based on fractal block coding. The first frame of the sequence was intra-frame coded, using a normal fractal image coding method (such as the Jacquin scheme). Every subsequent frame thereafter was coded using domain blocks constructed from the previous decoded frame. Two types of transformations were defined:

a) Carry forwards – This is a type of motion compensation, where a matching block of the same size was found from the previous decoded frame. Here, the transformation was just a translation.

b) Fractal codes – The transformation was contractive and a matching transformed domain block of a larger size was found.

The decoding procedure is extremely fast and reported to have been achieved in real time. This was due to the inter-frame coding being causal and only a single application of the transformations was necessary to perform the decoding.

The results published claimed compression ratios of 79:1, 58:1 and 21:1 resulting in recovered PSNR of 30.8 dB, 32.6 dB and 39.2 dB respectively. The 8-bit greyscale sequence used was the "Shuttle Launch" sequence of size 160x120 per frame.

#### 3.3.1.2 Hürtgen and Büttgen

In 1993, Hürtgen and Büttgen [18] developed a fractal coding technique for low bit-rate video. Their technique applied motion estimation by the frame differencing method. Thereafter, they applied fractal transform techniques to code the motion portions. Intra-frame coding was applied, and both the domain and range blocks were constructed from the same frame, where the range blocks represented the motion regions. They also used a three level block splitting method in their algorithm. Since fractal image compression was performed, the decoding process is iterative.

It was reported that using the coder on the 352x288, greyscale "Miss America" sequence, bit-rates of 128 kbit/s, 64 kbit/s and 32 kbit/s were achieved, resulting in recovered PSNR of 36-37 dB, 34-35 dB and 30-32 dB respectively.

#### 3.3.1.3 LU and Yew

In 1996, LU et al [19] published a fractal video compression scheme based on the quadtree fractal block coder. The first frame of a given sequence was compressed using the quadtree fractal image compression method. For inter-frame coding, three types of blocks were defined:

a) Type one block – A block in the current frame is identical to the corresponding block in the previous frame.

b) Type two block – A block in the current frame is a translation of a block in the previous frame.

c) Type three block – A block in the current frame is an affine transformation of a block in the previous frame.

The type one and type two blocks provides some kind of motion estimation, motion compensation technique. Each range block is checked if it can be coded using a type one, type two or a type three block, respectively. If a match cannot be found, the range block is partitioned into four smaller range blocks, according to the quadtree method. The search process continues for each smaller range block. The reference frame used in this coding algorithm is based on the hierarchy of a video sequence. Since inter-frame coding is performed, only a single iteration is required in the decoding procedures.

In the reported experiments on the "Salesman" sequence of size 256x256, a compression ratio of 57.9 was achieved yielding a PSNR of 29.17 dB.

### 3.3.2 Three Dimensional Fractal Coders

Three dimensional fractal coding or 3-D IFS is a direct extension of Jacquin's 2-D fractal block coding scheme. The difference between 2-D and 3-D coding is that image signals are bounded in all directions, but video sequences are only bounded in the spatial dimension and infinite in the temporal dimension [21]. Thus 3-D coding involves the temporal partitioning of the video frames into groups of pictures (GOP) as shown in Figure 1-1. Each GOP is then coded as a still image, by replacing 2-D square blocks with 3-D cubes. Hence the range and domain blocks are now in the form of cubes.

As in normal fractal compression, range cubes must be smaller than domain cubes. Hence geometric contractions must be performed on domain cubes in both the spatial and temporal directions. However, temporal redundancies consist of frame to frame similarities that occur on the same scale. Thus temporal contraction is not a requirement and is removed in the majority of 3-D fractal coders. This consequently implies that range and domain cubes be of the same size in the temporal direction.

The contracted domain cube must then be transformed, using contractive affine transformations as in the 2-D case. For 3-D cubes, more pixel shuffling operations are possible. However, more transformations taken results in increased computational complexity of the algorithm. For simplicity, many authors have decided to use the eight spatial isometry operations (defined in Jacquin's coding method) as well as two temporal operations ([23] and [24]). The temporal operations are the time reverse, and the identity isometries.

On computing the above, the best matching transformed domain cube is sought for each of the range cubes (Figure 3-15). If the matching criterion is not met, a similar approach to the quadtree partitioning method is implemented. Here, the range cube split can be performed either temporally or spatially [24]. These methods are detailed in Figure 3-16.

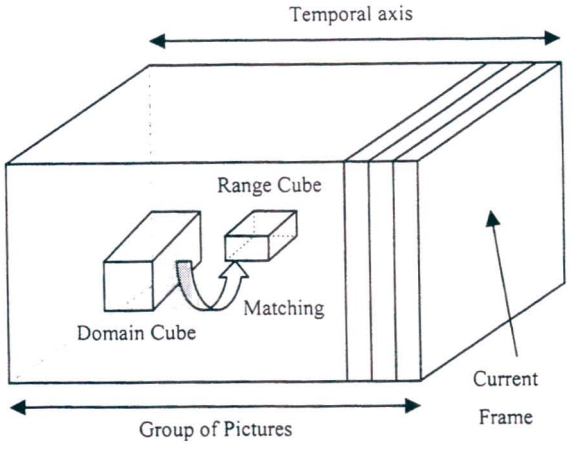



Figure 3-15: Matching between 3-D cubes in a GOP [23]

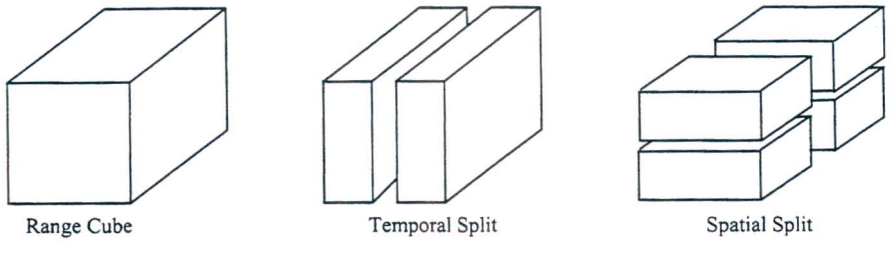



Figure 3-16: Spatial split versus temporal split of a range cube

Reusens [20] based his matching criteria on the MSE. If a match was not found, both partitioning methods were employed, with the better of the two selected. Lazar and Bruton [21] used a similar partitioning method to Reusens, but their choice of split was based on the error distribution within the range block. To account better for areas of motion, a new way of domain cube construction was introduced by Barakat and Dugelay [23]. It consisted of performing spatial shifting in consecutive frames of the domain cubes, with prefixed amplitudes in the x and y directions. However, this construction was only used in areas where the normal block matching was poor.

The decoding process of the 3-D fractal scheme is also iterative, within each GOP. Lazar and Barakat allowed domain cubes to be constructed outside the GOP and hence their coding scheme only required one iteration decoding.

#### 3.3.2.1 Performance of 3-D Fractal Coders

In general, 3-D block coding techniques have the potential for higher compression ratios than intra/inter frame coders. As with fractal compression, at high compression ratios, 3-D coders show severe blocking (or tiling) artefacts. It has been shown that a compression ratio of 250 has been obtained on the "Miss America" sequence, resulting in a decoded average PSNR of 32 dB [24]. However the complexity of 3-D coders is significantly higher than the intra/inter frame coders.

#### 3.3.2.2 Complexity of 3-D Fractal Coders

Ideally, domain cubes could be taken from anywhere in the sequence, but this is not practical. Thus the domain cube constructions are restricted to the GOP. Also, domain cubes could be overlapping as in 2-D fractal coders. However, the use of more domain cubes will increase the memory requirement of the system, making it impractical. Hence for all the proposed coding schemes, the domain cubes are non-overlapping and constrained to a GOP. In implementations of the 3-D scheme, the search is also limited to the neighbourhood of the range cube where finding a match is most likely.

The time and memory requirement for computing the transformations on domain cubes is significant. Moreover, the choice of range block partitioning requires significant resources, since each split is performed and the best split is chosen. Therefore, for these reasons, the investigation of 3-D coders will not be considered any further, and is beyond the scope of this project.

## 3.4 Summary

The majority of fractals are generated by IFS. The most important theorem in fractal theory is the contraction mapping principle. This theorem is fundamental in the production of fractals through IFS and guarantees a convergence to a unique attractor. Fractal compression is the inverse of fractal generation. The collage theorem provides a solution to this inverse problem. This theorem states that in order to find an IFS whose attractor looks like the given image, a set of contractive transformations must be found, such that the "distance" between the given image and the transformations on that image is small.

The first fractal compression methods attempted to reproduce an image by a combination of fractals, whose IFS was known. This method proved unsuccessful since there was no automated encoding algorithm. Fractal compression became a reality by the introduction of Jacquin's fractal block coding system. This scheme takes advantage of local self-similarity present in images i.e. a part of an image is similar to another part of that same image. Hence, the coding algorithm involved dividing the image into blocks of a fixed size, and for each block, a properly transformed copy of another block in the same image must be found that matches the block in question. The original image is regenerated by iterating each transformation obtained at the encoding stage, to its attractor.

One of the main features of fractal coding is the large encoding times. This is due to the block searching method employed. But on the other hand, the decoding process is much faster. The compression ratio of fractal systems can be increased by using larger block sizes. However, this has a direct negative effect on the recovered image quality. The R-D performance of fractal block coders can be improved by considering other partitioning schemes. A comparison of these different methods reveals that the horizontal-vertical partitioning method operates efficiently. All fractal block coders produces a tiling (or blocking) artefact in the decompressed image, especially at high compression ratios.

# CHAPTER 4 - WAVELETS AND THE WAVELET TRANSFORM

Wavelet analysis of images arose due to the fact that the wavelet transform was able to approximate a smooth function, with a small number of basis elements. Images were thought of as locally smooth functions that could be well modelled as piecewise functions. The downfall of other transform based systems is that the transform did not completely decorrelate all pixels in the image and neither did it provide the necessary energy compaction of the pixels. Theses two properties are vital in implementing compression systems. For example, the Fourier transform bases are very exact in frequency, but are not spatially precise. In other words, the energy of the Fourier coefficients are concentrated in one frequency, but are spread over all space. However, the wavelet coefficients have fairly good frequency concentration as well as spatial compactness. For example, if the image edges are not too closely packed, most wavelet coefficients will not intersect with them, thus performing a better decorrelation. Moreover, the frequency spectrum is partitioned finely at low frequency and coarsely at higher frequency, to obtain maximum benefits for a fixed number of basis elements. We can further motivate the use of the wavelet transform in image coding by using the notion of multi-resolution analysis.

Section 4.1 presents a brief introduction to wavelets and the wavelet transform. It is here where the wavelet transform of an image is defined, and methods to produce the transform are described. Section 4.2 discuses the reasons for implementing the wavelet transform in compression algorithms. Several statistical properties for the wavelet transform are provided. The following section (Section 4.3) provides a literature survey of the most effective wavelet based image coders developed. These coders vary in the quantization and entropy coding stages of transform coder paradigm. For each algorithm, the important concepts are covered.

## 4.1 Wavelet Theory

### 4.1.1 Expansion Functions and Multi-Resolution Analysis (MRA)

**Definition 4-1: Expansion Function**: Given a continuous, square integrable function $f(x)$, $f(x)$ can be expanded (or approximated) by:

$$f(x) = \sum_k \alpha_k \varphi_k(x), \tag{4.1}$$

where $\varphi_k$ is the expansion function and $\alpha_k$ the expansion coefficients [40].

Hence, in Equation (4.1), $f(x)$ is written as a linear combination of expansion functions and expansion coefficients.

**Definition 4-2: Basis Function**: If the expansion in Equation (4.1) above is unique, i.e. there is only one set of $\alpha_k$ for any function $f(x)$, then $\varphi_k(x)$ are called basis functions, and the expansion set, $\{\varphi_k(x)\}$ is known as the basis [40].

**Definition 4-3: Function Space**: A function space $V$, which is referred to as the closed span of the expansion set, $\{\varphi_k(x)\}$, is denoted as

$$V = \overline{\underset{k}{Span}\{\varphi_k(x)\}}. \tag{4.2}$$

So, if $f(x) \in V$, it means that $f(x)$ falls in the closed span of $\{\varphi_k(x)\}$ and can be expanded in the form of Equation (4.1) [40].

**Definition 4-4: Dual Function**: For any function space $V$ and corresponding expansion set $\{\varphi_k(x)\}$, there is a set of dual function, denoted as $\{\tilde{\varphi}_k(x)\}$, that can be used to compute the expansion coefficients $\alpha_k$, for any $f(x) \in V$ [40]. The expansion coefficients can be calculated as follows:

$$\alpha_k = \int \tilde{\varphi}_k^*(x) f(x) dx, \tag{4.3}$$

where $\varphi_k^*$ denotes the conjugate of $\varphi_k$.

**Theorem 4-1: Orthonormal Basis**: If the expansion functions form an orthonormal basis for $V$, the basis and its dual are equivalent. In mathematical term, if

$$\int \varphi_j^*(x) \tilde{\varphi}_k(x) dx = \begin{Bmatrix} 0 & j \neq k \\ 1 & j = k \end{Bmatrix}, \text{ then } \varphi_k(x) = \tilde{\varphi}_k(x). \tag{4.4}$$

**Definition 4-5: Integer translated and binary scaled expansion function**: We can vary the resolution of the approximations by contracting and dilating the expansion function. This is achieved by letting the expansion function be composed of integer translations and binary scalings of the function $\varphi(x)$, i.e.

$$\varphi_{j,k}(x) = 2^{j/2} \varphi(2^j x - k), \tag{4.5}$$

for all $j,k \in \mathbb{Z}$ (set of integers) and $\varphi(x) = L^2(\mathbb{R})$ (set of measurable, square-integrable one dimensional functions) [40]. Here, $k$ determines the position of $\varphi(x)$ along the x-axis (integer

translation), $2^j$ controls how broad or narrow $\varphi(x)$ is along the x-axis, and $2^{j/2}$ controls its height or amplitude.

If we now restrict $j = j_0$, where $j$ represents the resolution, we obtain

$$\varphi_{j_o,k}(x) = 2^{j_o/2}\varphi(2^{j_o}x - k), \tag{4.6}$$

with the subspace defined as

$$V_{j_o} = \overline{\underset{k}{Span}\{\varphi_{j_o,k}(x)\}}. \tag{4.7}$$

If $f(x) \in V_{j_o}$, it can be written as,

$$f(x) = \sum_k c_{j_o}(k)\varphi_{j_o,k}(x), \tag{4.8}$$

where $c_{j_o}$ are the expansion coefficients for each integer translation $k$.

**Example 4-1**: Suppose the expansion function is the unit height, unit width scaling function (also known as the Haar function). The function is defined as:

$$\varphi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & otherwise \end{cases}. \tag{4.9}$$

Figure 4-1 (a) to (d) show four of the many expansion functions that can be obtained by substituting Equation (4.6) in this Haar function. Note that as $j$ is increased, the resulting expansion functions are narrower and closer together. Figure 4-2 illustrates a signal being approximated by the Haar function. As can been seen, the higher the resolution, the better the approximation of the signal becomes. However, more coefficients are used to define higher resolutions than the lower ones, since the translations will be smaller.

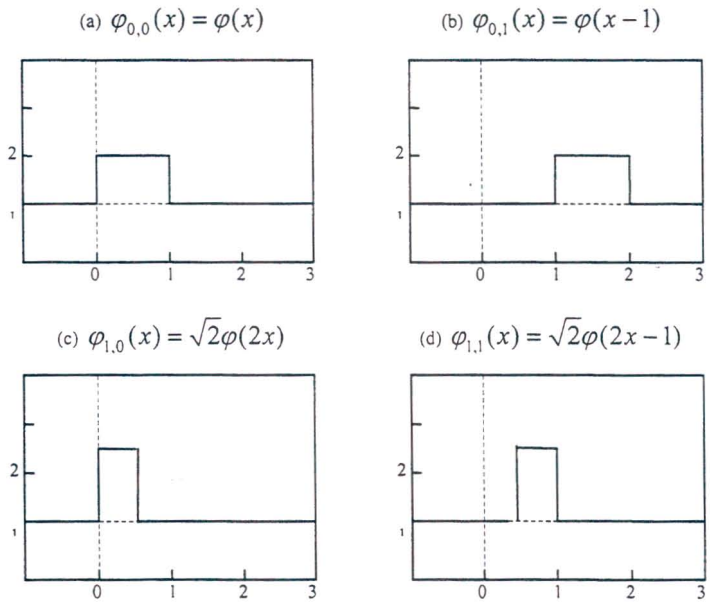



Figure 4-1: Haar scaling functions

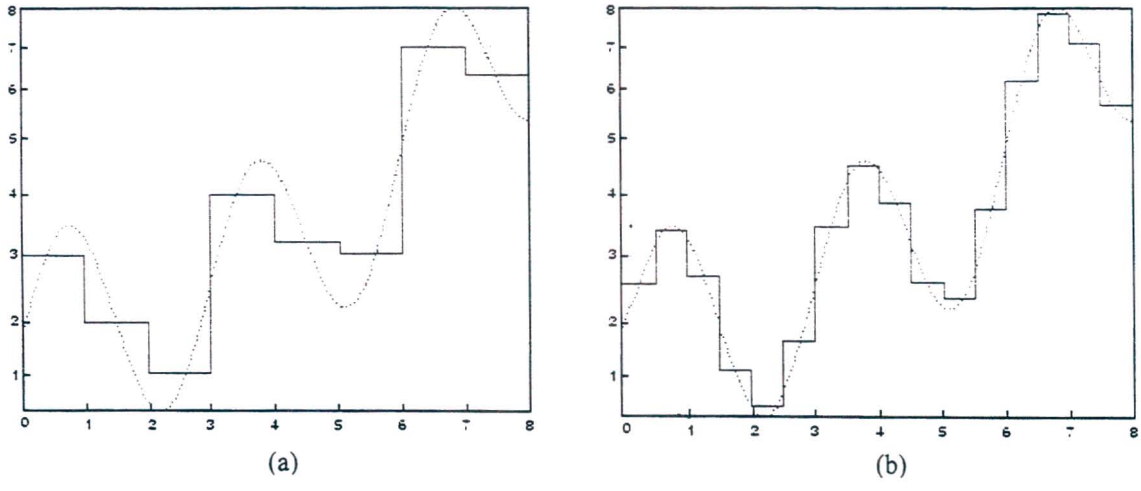



Figure 4-2: A continuous function $f(x)$ (plotted as dotted line) is approximated by the Haar function (solid line) (a) Resolution $j$=0 (b) Resolution $j$=1

An important property of multi-resolution analysis, is that higher resolution functions must also contain lower resolution functions [40], i.e.

$$V_{-\infty} \subset \cdots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots \subset V_{\infty}. \qquad (4.10)$$

The above Equation (4.10), indicates that if $f(x) \in V_0$, $f(x)$ must be $\in V_1$, since $V_0 \subset V_1$ ($V_0$ is a subspace of $V_1$).

**Theorem 4-2: Multi-Resolution Equation**: The refinement equation (or multi-resolution analysis equation),

$$\varphi(x) = \sum_n h_\varphi(n)\sqrt{2}\varphi(2x-n), \tag{4.11}$$

states that the expansion functions of any subspace can be built from double-resolution copies of themselves i.e. from expansion functions of the next higher resolution space [40]. The $h_\varphi(n)$ coefficients in this recursive equation are called the scaling function coefficients, and $h_\varphi$ is referred to as the scaling vector.

### 4.1.2 Wavelets

Instead of approximating a function by using different resolution approximation for different parts of the function, we can approximate a function using the lowest resolution and thereafter adding higher resolution to each lower resolution approximation, to improve the quality of the approximations. This is where wavelets fit in.

**Definition 4-6: Wavelet Function**: The wavelet function $\psi(x)$ spans the difference between any two adjacent scaling subspaces $V_j$ and $V_{j+1}$ [40]. We define the wavelet function as

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k), \tag{4.12}$$

and its subspace,

$$W_j = \overline{\underset{k}{Span}\{\psi_{j,k}(x)\}}. \tag{4.13}$$

In other words, $V_j$ is the lower resolution space and $W_j$ adds more resolution to bring the resolution to the $V_{j+1}$ space. Hence $V_2 = V_0 \cup W_0 \cup W_1$, where $V_1 - V_0 = W_0$, $V_2 - V_1 = W_1$ and $V_0$ is the lowest resolution subspace. Note that $\cup$ represents the union of subspaces. So, $f(x)$ can now be written as,

$$f(x) = \sum_k c_{j_o}(k)\varphi_{j_o,k}(x) + \sum_{j=j_o}^{\infty}\sum_k d_j(k)\psi_{j,k}(x). \tag{4.14}$$

The first term in Equation (4.14) represents the approximation of $f(x)$ at the base resolution $j_0$, hence $c_{j_o}$ is referred to as the approximation coefficients and $\varphi_{j_o,k}(x)$ the approximation function. The second term adds more detail to obtain $f(x)$. Detail is provided at higher resolutions, $j \geq j_0$. $d_j$ are the detail coefficients and $\psi_{j,k}(x)$ the detail functions. Equation (4.14) is commonly known as the Wavelet Series Expansion.

The wavelet function, $\psi(x)$, can be built from the scaling function, $\varphi(x)$, by using the MRA Equation (4.11). This is due to the fact that the wavelet space resides in the space spanned by the next higher resolution scaling function. Hence,

$$\psi(x) = \sum_n h_\psi(n)\sqrt{2}\varphi(2x-n), \tag{4.15}$$

where $h_\psi(n)$ are the wavelet coefficients.

### 4.1.3 The Wavelet Transform

In order to calculate the wavelet transform of a signal $f(x)$, we need to find both the approximation and wavelet coefficients that are used to approximate the signal. Like the Fourier transform, the DWT of a function is expanded as samples of a continuous function $f(x)$. The DWT is given by,

$$\begin{aligned} W_\varphi(j_o,k) &= \frac{1}{\sqrt{M}}\sum_{x=0}^{M-1} f(x)\tilde{\varphi}_{j_o,k}(x) \\ W_\psi(j,k) &= \frac{1}{\sqrt{M}}\sum_{x=0}^{M-1} f(x)\tilde{\psi}_{j,k}(x) \end{aligned}, \tag{4.16}$$

for $j \geq j_0$. Here, $f(x)$, $\varphi_{j_o,k}(x)$ and $\psi_{j,k}$ are functions of the discrete variable $x = 0,1,2,\ldots,M-1$. On computation of the above formulas, $W_\varphi(j_o,k)$ generates approximation coefficients at resolution scale $j_0$, and $W_\psi(j,k)$ generates detail coefficients at each of the resolution scales $j_0, j_1, j_2, \ldots$. Normally, we let $j_0 = 0$ and select $M = 2^J$ (J = highest resolution scale, M = number of samples). Therefore, $j = 0,1,2,\ldots,J-1$ and $k = 0,1,2,\ldots,2^{j-1}$. To find the inverse DWT, we use the following equation [40],

$$f(x) = \frac{1}{\sqrt{M}}\sum_k W_\varphi(j_o,k)\varphi_{j_o,k}(x) + \frac{1}{\sqrt{M}}\sum_{j=j_o}^{\infty}\sum_k W_\psi(j,k)\psi_{j,k}(x). \tag{4.17}$$

#### 4.1.3.1 Fast Wavelet Transform

A computationally efficient method of implementing the DWT is given by the Mallat-Herringbone algorithm [42] and is known as the fast wavelet transform (FWT). The FWT resembles the two-band subband coding scheme shown in Figure 4-3, where $W\varphi(j+1,n)$ are the DWT approximation coefficients at scale $j+1$. The DWT approximation and detail coefficients at scale $j$, are then computed by the convolution of the DWT approximation coefficients at scale $j+1$, with the time-reversed scaling and wavelet vectors, $h_\varphi(-n)$ and $h_\psi(-n)$ respectively, and down sampling the result by 2, i.e.

$$W_{\varphi}(j,k) = h_{\varphi}(-n) * W_{\varphi}(j+1,n)\big|_{n=2k,k\geq 0}$$
$$W_{\psi}(j,k) = h_{\psi}(-n) * W_{\varphi}(j+1,n)\big|_{n=2k,k\geq 0} \qquad (4.18)$$

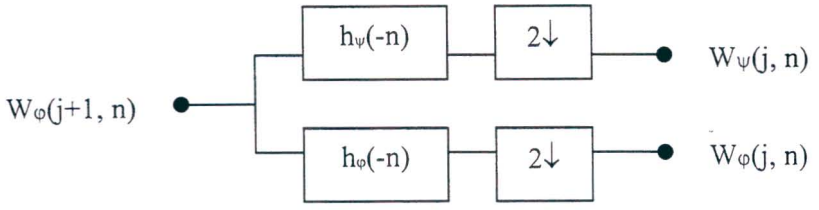



Figure 4-3: One stage FWT analysis bank [40]

The above filter bank in Figure 4-3 has the effect of splitting the subspace spanned by $W_{\varphi}(j+1,n)$, i.e. $V_{j+1}$, into a high pass detail component $W_j$ (spanned by $W_{\psi}(j,n)$) and a low pass approximation component $V_j$ (spanned by $W_{\varphi}(j,n)$). This effect is shown below in Figure 4-4.

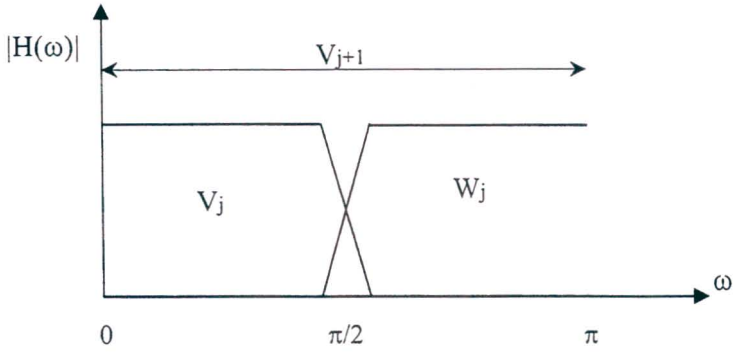



Figure 4-4: Frequency splitting characteristics of a one stage analysis bank [40]

The structure of Figure 4-3 can be extended to multistage structures for computing the DWT at two or more successive scales. Figure 4-5 details a two–stage filter bank.

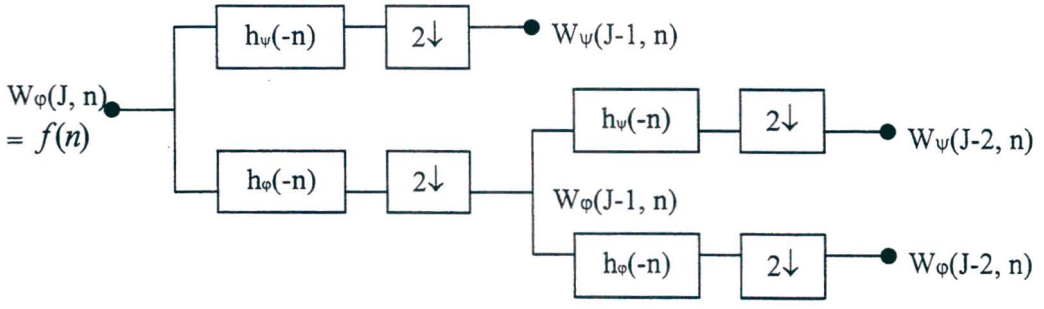



Figure 4-5: Two-stage FWT analysis bank [40]

Normally, the highest resolution scale coefficients are assumed to be samples of the function $f(x)$, i.e. $W_{\varphi}(J,n) = f(n)$, where $J$ is the highest resolution scale. Hence, $f(x) \in V_J$. As shown

in Figure 4-6, the first filter bank splits $f(x)$ into a high pass detail component, corresponding to coefficients $W_\psi(J-1,n)$ (subspace $W_{J-1}$) and a low pass approximation component, which corresponds to coefficients of $W_\varphi(J-1,n)$ (subspace $V_{J-1}$). The second filter bank splits the subspace $V_{J-1}$ into smaller subspaces $W_{J-2}$ and $V_{J-2}$, corresponding to DWT coefficients $W_\psi(J-2,n)$ and $W_\varphi(J-2,n)$ respectively.

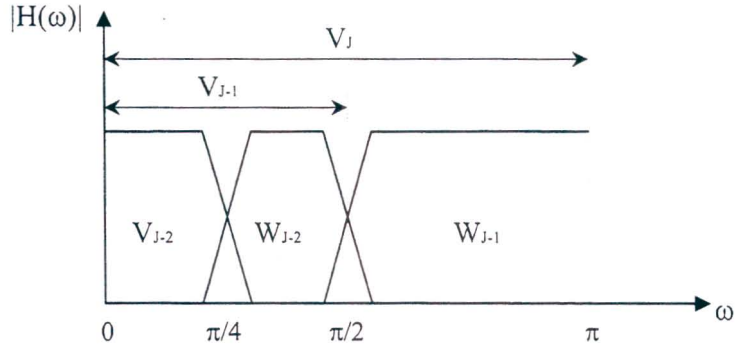



Figure 4-6: Frequency splitting characteristics of a two stage analysis bank [40]

An inverse fast wavelet transform ($FWT^{-1}$) can also be achieved by noting the similarity between the FWT and the two-band subband decoding system. The $FWT^{-1}$ synthesis filter bank appears in Figure 4-7, below.

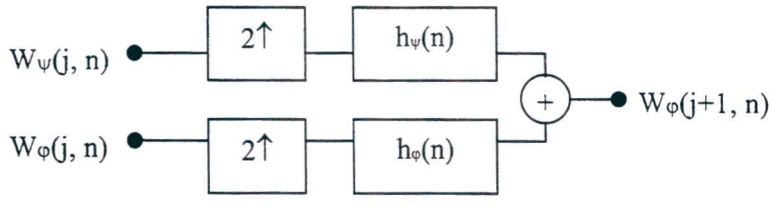



Figure 4-7: The $FWT^{-1}$ synthesis filter bank

The above $FWT^{-1}$ filter bank implements the computation

$$W_\varphi(j+1,k) = h_\varphi(k) * W_\varphi^{up}(j,k) + h_\psi(k) * W_\psi^{up}(j,k)\big|_{k\geq 0} \qquad (4.19)$$

where, $W^{up}$ refers to up sampling by 2 (i.e., inserting zeroes between elements of $W$ so that it is twice the original length).

Figure 4-8 provides three different representations of sampled data. The standard time domain provided instants when events occur, but no frequency information. The Fourier transform, on the other hand pinpoints the frequencies that are present in events that occur over long period of time, but provides no time resolution. The wavelet transform addresses this problem, and provides a time-scale representation of the events. Each tile in the DWT represents an equal

portion of the time-frequency plane. Long time intervals are provided, where precise low frequency information is required, and narrower tiles are provided, when high frequency information is needed.

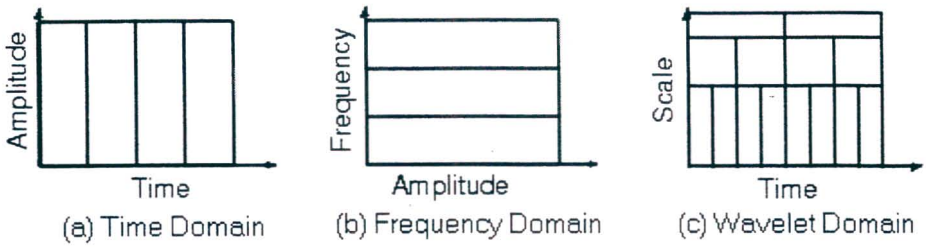



Figure 4-8: (a) Time based representation of sampled data, (b) Frequency based - DFT of data, (c) DWT of data

### 4.1.3.2 Two Dimensional Wavelet Transform

For a two dimensional function *f(x,y)* (like an image), the function depends on two variables *x* and *y*. Hence we need to define two dimensional scaling and wavelet functions, where each is a product of the one dimensional scaling ($\varphi$) and wavelet ($\psi$) functions. The four products that are produced are:

(i) $\varphi(x,y) = \varphi(x)\varphi(y)$ = separable scaling function

(ii) $\psi^H(x,y) = \psi(x)\varphi(y)$ = separable horizontal directional wavelet function, which measures variations along columns.

(iii) $\psi^V(x,y) = \varphi(x)\psi(y)$ = separable vertical directional wavelet function, which measures variations along rows.

(iv) $\psi^D(x,y) = \psi(x)\psi(y)$ = separable diagonal directional wavelet function, which corresponds to variations along diagonals.

The scaled and translated two dimensional basis functions are defined as [40]:

$$\begin{aligned} \varphi_{j,m,n}(x,y) &= 2^{j/2}\varphi(2^j x - m, 2^j y - n) \\ \psi^i{}_{j,m,n}(x,y) &= 2^{j/2}\psi^i(2^j x - m, 2^j y - n) \end{aligned}, \tag{4.20}$$

for $i = H, V, D$ (each directional wavelet function).

Hence the DWT of a function *f(x,y)* at scale $j_o$ is defined as [40]:

$$\begin{aligned} W_\varphi(j_o,m,n) &= \frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\varphi_{j_o,m,n}(x,y) \\ W^i{}_\psi(j,m,n) &= \frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\psi^i{}_{j,m,n}(x,y) \end{aligned}, \tag{4.21}$$

for $i = H, V, D$ and MxN is the size of $f(x,y)$. $W_\varphi(j_o,m,n)$ are the approximation coefficients at scale $j_0$, while $W^i_\psi(j,m,n)$ are the horizontal, vertical and diagonal coefficients at scales $j \geq j_0$.

The two dimensional DWT can easily be computed using the one dimensional FWT method. To compute the fast two dimensional wavelet transform, we compute the one dimensional FWT of the rows of $f(x,y)$, followed by the one dimensional FWT of the resulting columns. This is shown in Figure 4-9. Further decompositions can be achieved by computing the DWT on the $W_\varphi(j,m,n)$ coefficients.

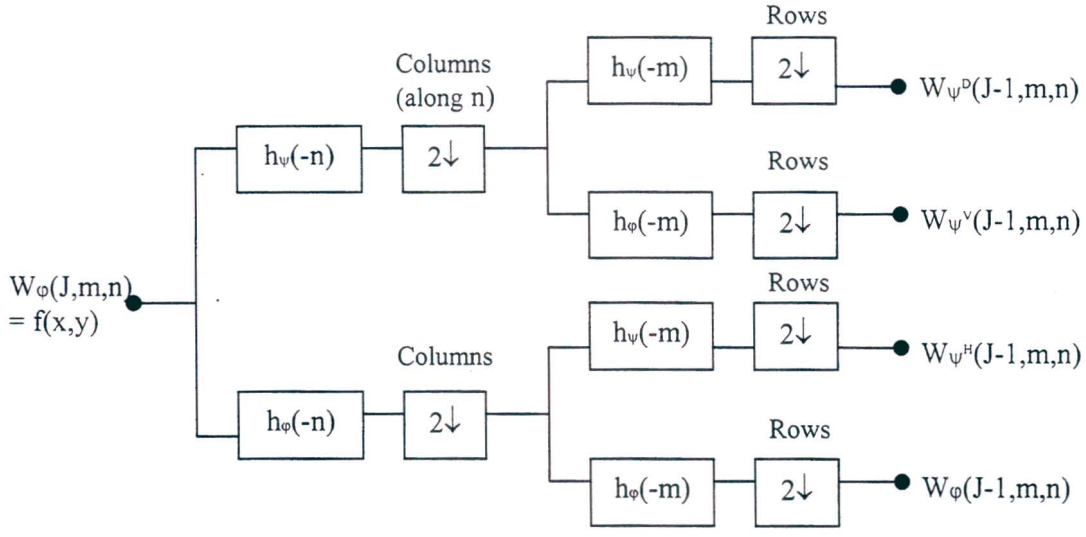



Figure 4-9: The two dimensional fast wavelet transform [40]

The transform coefficients $W_\varphi$, $W_\psi^H$, $W_\psi^V$ and $W_\psi^D$ correspond to different subband images namely the LL, LH, HL and the HH subband respectively. This is depicted in the centre diagram of Figure 4-10(a). Figure 4-10(b) shows a two level wavelet decomposition of an image.

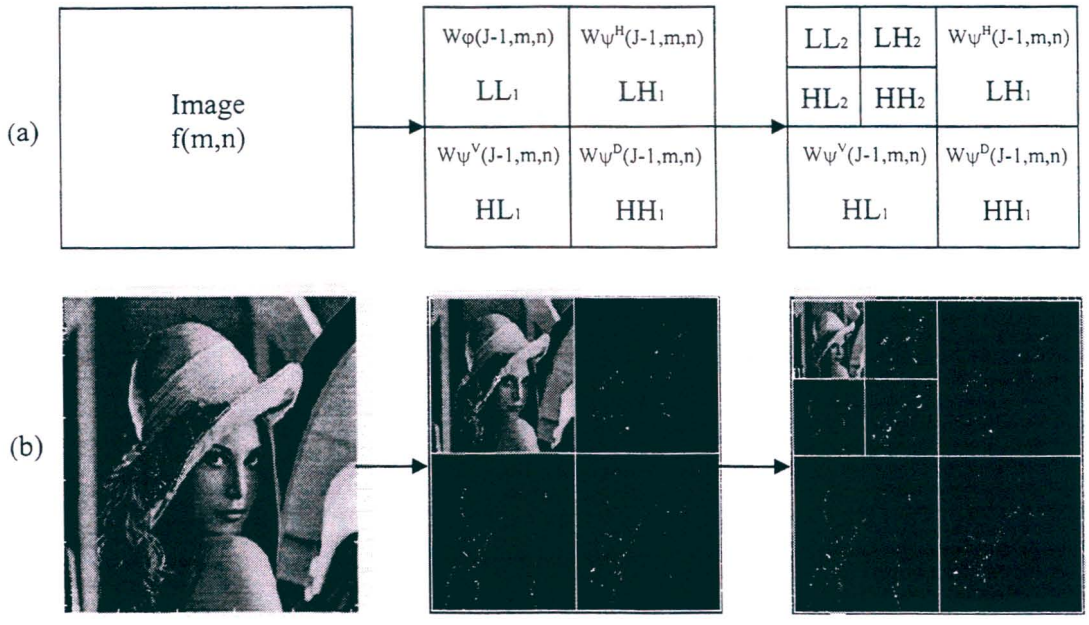



Figure 4-10: Two level wavelet decomposition of:
(a) any two dimensional function, (b) "Lena" image

The inverse FWT of two dimensional functions can be computed using Figure 4-11.

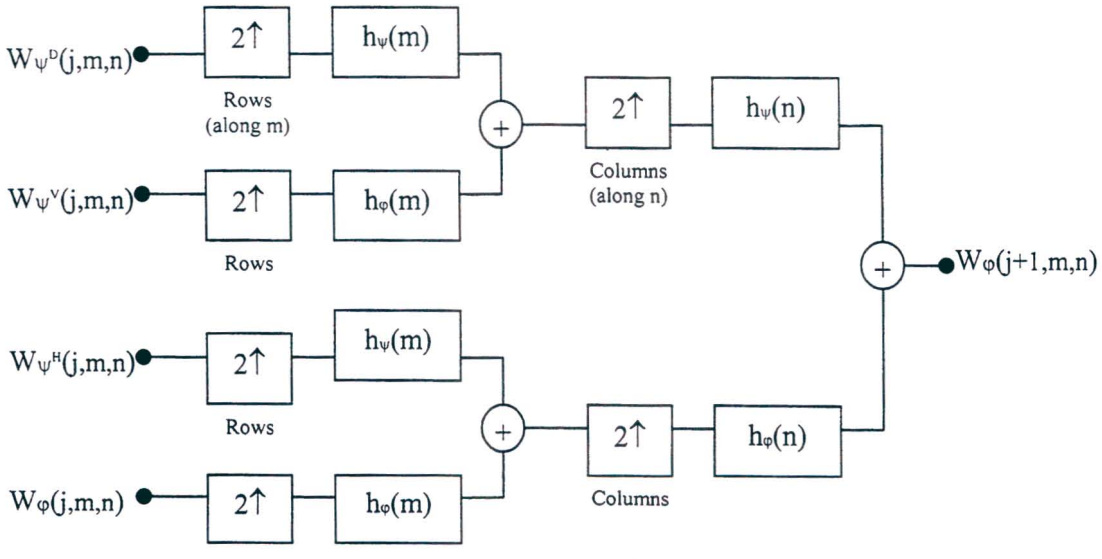



Figure 4-11: Inverse wavelet transform of a two dimensional function [40]

## 4.2 Wavelet Basis and Properties Suited For Image Coding

### 4.2.1 Construction of Wavelet Basis

The key question to ask is how to obtain basis scaling ($\varphi$) and wavelet ($\psi$) functions. These functions need to satisfy the four fundamental requirements of MRA [40] viz.

(i) The scaling function is orthogonal to its integer translates i.e. the inner product of the function with its integer translates is zero.

(ii) The subspaces spanned by the scaling function at low scales are nested within those spanned at higher scales i.e. $V_{-\infty} \subset \cdots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots \subset V_{\infty}$.

(iii) The only function that is common to all $Vj$ is $f(x) = 0$.

(iv) The scaling function can represent any function with precision.

There are also other restrictions placed on the basis function when performing image compression. These are

a) Smoothness of basis functions [39] – Natural images tend to contain locally smooth regions, thus in lossy wavelet image compression, it is of vital importance that the basis functions be smooth. If the wavelets contain discontinuities, any quantization errors in the coefficients, will appear as a linear combination of the wavelet basis functions $\varphi(x)$ and $\psi(x)$ in the decompressed image. These discontinuities will cause the decompressed image to contain highly visible unacceptable artefacts, particularly in smooth regions of the image.

b) Accuracy of approximation – Wavelets can be used to reproduce any polynomial up to a given degree. If a continuous function $f(x)$ can be represented by a polynomial of a certain degree, we can reproduce this function exactly with just a few wavelet coefficients. The number of vanishing moments of the dual wavelet $\tilde{\psi}(x)$ determines the degree of polynomial that can be reproduced.

**Definition 4-7: Vanishing moments**: A function $f(x)$ has $p$ vanishing moments if

$$\int_{-\infty}^{\infty} x^n f(x)dx = 0 \quad for \quad 0 \le n \le p \tag{4.22}$$

So if a wavelet basis of $L^2(\mathbb{R})$, generated by $\psi(x)$ and dual wavelet $\tilde{\psi}(x)$, has N vanishing moments, it can reproduce a polynomial of degree N-1 [8].

c) Size of the support of wavelet basis – If a function $f(x)$, equal to a polynomial of degree N-1, is reproduced by the dual wavelet $\tilde{\psi}(x)$ with N vanishing moments, then the wavelet basis function $\psi(x)$ will have a zero coefficient wherever $\tilde{\psi}(x)$ lies completely in the region where $f(x)$ is a polynomial. The smaller the support of $\tilde{\psi}(x)$, the more zero coefficients are produced. Edges normally produce large wavelet coefficients. Thus if the $\tilde{\psi}(x)$ is very large, the more likely it is to overlap and edge. Therefore it is vital that the wavelets have a reasonably small support.

In the design of a wavelet basis, there is a trade-off between the accuracy of approximation, smoothness and support. If the support is too small, the number of vanishing moments reduces, hence the accuracy and regularity suffers. On the other hand, the support can be increased; producing a high degree of smoothness and a large number of vanishing moments, but a large number of non-zero wavelet coefficients will be produced. There is extensive literature present that deals in the design of wavelet basis with particular properties given in [39] and [8].

### 4.2.2 Properties Used in Compression Algorithms

a) Trends and Anomalies – Images consist mainly of areas of spatial trends (high statistical spatial correlation) and few anomalies (edges and object boundaries). Therefore in compressing images, the majority of the bits must be allocated to the trends, rather than to the anomalies. Trends are localized in the frequency domain, while it endures a large number of lags in the time domain, whereas anomalies tend to be wideband in the frequency domain and localized in the time domain. The advantage of wavelet theory is that both anomalies and trends can be analysed on an equal statistical footing. The wavelet transform allocates some coefficients to high data lags corresponding to narrowband low frequency range and some coefficients to short data lags corresponding to a wideband high frequency range.

b) Statistical independence – Transform based coders rely on the transform to completely decorrelate the coefficients for pixel quantization errors not to affect the decoded image significantly. The wavelet transform coefficients are shown to be almost statistically independent; therefore samples do not depend on each other. Hence if the inverse transform is calculated, it does not matter if the transform coefficients are exact or not.

c) Insignificant coefficients – A good transform coder is one that provides good energy compaction of the pixel values. The wavelet transform produces a number of coefficients that have magnitudes close to zero. This property is exploited in a number of compression algorithms. If the locations of these coefficients are implicitly known, then these coefficients need not be coded. To illustrate this property, Figure 4-12 represents the statistical representation of a three level wavelet transformed "Lena" image. As shown in this figure, the majority of the coefficients of the detail component of all levels are clustered around zero. Furthermore, the distribution of coefficients in the detail components increases from the lowest level of the transform.

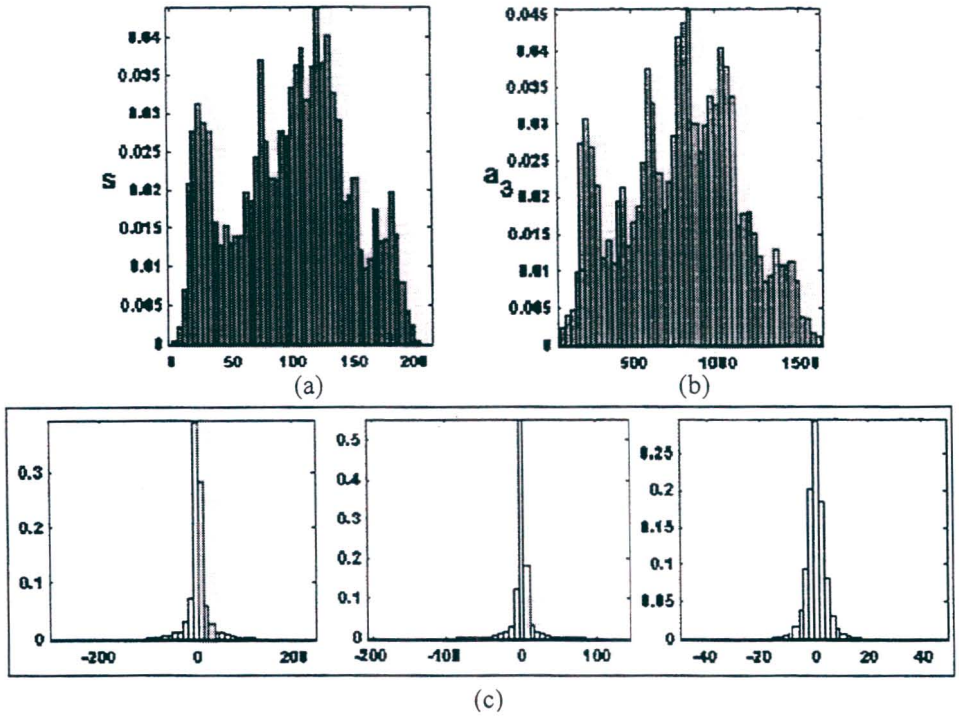



Figure 4-12: Statistical distribution of a three level wavelet transformed "Lena" image
(a) Distribution of original image (b) Distribution of approximation component
(c) Distribution of horizontal detail component at level 3, 2, and 1 from left to right

### 4.2.3 Choice of Wavelet Basis

The choice for the optimal wavelet basis for image compression is difficult. As explained in Section 4.2.1, there are a number of design decisions to be made. In the extension for the computation of the 2-D wavelet transform (Section 4.1.3.2), separable wavelet basis were formed as a combination of scaling and wavelet functions. The use of separable wavelet transform reduces the problem of designing efficient wavelets.

A test was performed using different wavelet basis functions in a wavelet image compression algorithm. The results were presented by Saha [43] in which several different images were compressed to a ratio of 16:1. The results are tabulated below in Table 4-1. According to the results, the best performing wavelet is the Daubechies 9/7 wavelet (CDF 9/7 in the table).

| Image Statistics | Lena | Barbara | Baboon | Peppers | Bengali | mri | nervecell |
|---|---|---|---|---|---|---|---|
| Mean | 99 | 117 | 129 | 120 | 213 | 57 | 117 |
| Median | 97 | 117 | 130 | 121 | 255 | 44 | 98 |
| Std. Dev. | 53 | 55 | 42 | 54 | 78 | 54 | 84 |
| Variance | 2796 | 2982 | 1789 | 2894 | 6116 | 2900 | 7001 |
| | | | | | | | |
| Wavelets | | | | | | | |
| Haar | 31.51 | 27.23 | 24.19 | 33.04 | 27.3 | 29.42 | 30.66 |
| Daub2 (4 coeff) | 33.03 | 28.32 | 24.77 | 34.37 | 24.92 | 31.59 | 32.4 |
| Daub4 (8 coeff) | 33.48 | 29.1 | 25.04 | 34.74 | 24.25 | 32.31 | 32.73 |
| Daub8 (16 coeff) | 33.65 | 29.64 | 25.1 | 34.38 | 22.77 | 32.15 | 33.04 |
| Adelson (9 coeff) | 33.93 | 29.51 | 25 | 34.78 | 24.66 | 32.38 | 33.6 |
| CDF-9/7 | 34.28 | 29.54 | 25.05 | 35.19 | 24.71 | 32.77 | 33.96 |
| (CDF-7/9) | 33.1 | 28.76 | 24.47 | 34.36 | 24.58 | 31.39 | 32.83 |
| CDF-9/11 | 33.97 | 29.8 | 24.68 | 34.68 | 24.21 | 32.28 | 33.46 |
| Odegard-9/7 | 34.3 | 30.16 | 24.98 | 35.08 | 24.62 | 32.73 | 33.9 |
| Brislawn-10/10 | 33.68 | 29.07 | 24.25 | 34.53 | 23.07 | 32.63 | 33.71 |
| Villasenor-10/18 | 34.15 | 30.09 | 25.33 | 35.19 | 23.21 | 32.85 | 33.96 |
| (Villasenor-18/10) | 33.41 | 29.2 | 24.65 | 34.39 | 24.36 | 31.67 | 32.85 |
| Villasenor-13/11 | 34.28 | 30.18 | 24.86 | 34.99 | 24.47 | 32.67 | 33.66 |
| (Villasenor-11/13) | 33.82 | 29.39 | 24.75 | 34.57 | 24.6 | 32.11 | 33.49 |
| Villasenor-6/10 | 34.04 | 29.57 | 24.57 | 35.16 | 23.85 | 32.73 | 33.72 |
| (Villasenor-10/6) | 33.14 | 28.75 | 24.5 | 34.15 | 24.84 | 31.26 | 32.61 |
| CDF-13/3 | 33.65 | 28.81 | 24.78 | 34.69 | 24.51 | 32.3 | 33.58 |
| (CDF-3/13) | 32.98 | 28.54 | 23.82 | 33.85 | 23.75 | 31.04 | 32.24 |
| Comp. Statistics | | | | | | | |
| Max | 34.3 | 30.18 | 25.33 | 35.19 | 27.3 | 32.85 | 33.96 |
| Min | 31.51 | 27.23 | 23.82 | 33.04 | 22.77 | 29.42 | 30.66 |
| Max Difference | 2.79 | 2.95 | 1.51 | 2.15 | 4.53 | 3.43 | 3.3 |
| CDF 9/7 Difference | 0.02 | 0.64 | 0.28 | 0 | 2.59 | 0.08 | 0 |

Table 4-1: Performance (PSNR) comparison between wavelet basis [43]

## 4.3 Wavelet Image Coding

Wavelet based compression algorithms, uses the transform coding setup (see Section 2.1). The first step is to perform the 2-D wavelet transform on the image. Section 4.2 outlines the procedure and choice of wavelet basis to perform the transform. This component performs the necessary decorrelation and energy compaction of the image samples. Different wavelet algorithms differ in the next two portions of the transform coder paradigm, namely the quantization and entropy coding steps.

Instead of quantizing the wavelet coefficients in a predetermined order of priority, different approaches are taken. This is made possible since the wavelet coefficients are spatially as well as spectrally compact. In parts of the image where the energy is spatially, but not spectrally compact, selection operators can be used to choose subsets of the wavelet coefficients that represent the signal efficiently. This is the essence of Zerotree coders, which will be discussed.

The first wavelet compression algorithm was the baseline compression method, explained by Davis, Nosratinia [39] and Mallat [42]. This algorithm first finds the position of significant transform values, i.e. coefficients that have magnitudes larger than a threshold $T_h$, via a specific scan order. The values are stored as a significance map:

$$sig(m) = \begin{cases} 0 & if\ |c(m)| < T_h \\ 1 & if\ |c(m)| \geq T_h \end{cases}, \tag{4.23}$$

where $m$ is the scanning index and $c(m)$ the coefficient of the wavelet transform at index $m$. Once the scan is completed, the significant values are encoded. Thereafter, the threshold is reduced and the process restarts. It was noted that a series of 0's followed by a series of 1's would be achieved. The general workings of this baseline method are used in the majority of state of the art wavelet compression algorithms which will be explained in the following sections.

### 4.3.1 Embedded Zerotree Wavelet

In 1993, Jerry Shapiro introduced the EZW algorithm [46]. EZW coding exploited the multi-resolution nature of the wavelet decomposition to give a completely new way of doing image coding. This algorithm was the first of the wavelet image encoders to efficiently capture the low frequency information (highest level of the wavelet transform or approximation component) as well as the localized high frequency information by the method of zerotree quantization. The EZW approach then set a new standard in modern wavelet image coding. At that time, this algorithm outperformed other existing coding standards at low bit-rates, thereby improving image compression.

**Definition 4-8: Embedded Coding**: Embedded coding is a process of encoding the transform magnitudes (or coefficients) such that it allows for progressive transmission of the compressed image.

Using an embedded code an encoder can terminate the coding at any point thereby allowing a target rate to be met exactly. The longer the process continues, more precision is added. The

EZW algorithm has the property that the bits in the bit stream are generated in order of importance, yielding a fully embedded code.

The EZW coder is based on two observations of the wavelet transform:

(i) In general, natural images have a lowpass spectrum. Hence the maximum and average absolute value of wavelet coefficients will get smaller as one moves from the lower frequency subbands (highest level) to the higher frequency subbands. This shows that progressive encoding is a very natural choice.

(ii) Large wavelet coefficients are more important than small coefficients, since they contain more information. Hence these coefficients are coded first.

It was noticed that the wavelet transform produces a natural tree-like structure, in that each subband in the wavelet transform contains different frequency information from the same spatial location. This inter-band structure representing the same spatial location looks like a tree (Figure 4-13). Each coefficient in the wavelet transform, at lower frequency subbands $(i,j)$, can be thought of as a parent node, having four children (or descendants) in the next higher subband at locations $(2i,2j)$, $(2i+1,2j)$, $(2i,2j+1)$ and $(2i+1,2j+1)$. Each of these children will in turn have four children in the next higher subband. Shapiro showed that although the wavelet transform removes the linear correlations between coefficients, there is still a strong correlation between the square of the coefficients and called this spatial self-similarity.

**Definition 4-9: Quadtree**: A quadtree is a tree of locations in a wavelet transform, with a root, its children and each of their children, and so on. These descendents of the root reach all the way back to the first level of the wavelet transform (Figure 4-13).

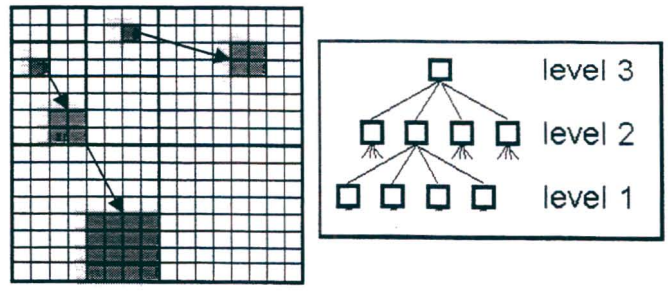



Figure 4-13: Parent-Child Dependencies of Quadtrees.
Two quadtrees are shown where the root of one is at level 3 and the other at level 2

**Definition 4-10: Zerotree**: A Zerotree is a quadtree in which all nodes are equal to or smaller than the root, which is smaller than the threshold against which the wavelet coefficients are currently being measured.

**Hypothesis 4-1: Zerotree Hypothesis**: The zerotree hypothesis states that if a wavelet coefficient $c$ at a coarse scale is insignificant with respect to a given threshold $T_h$, i.e. $|c|<T_h$ then all wavelet coefficients of the same orientation at finer scales are also likely to be insignificant with respect to $T_h$.

The Zerotree hypothesis is a direct result of the special self-similarity that exists in the wavelet transform. Zerotrees allow for concise encoding of the positions of significant values by creating a highly compressed description of the location of insignificant values. The tree is coded with a single symbol ('ZTR') and reconstructed by the decoder as a quadtree filled with zeroes. Zerotrees can be useful only if they occur frequently. Fortunately, the transformation of natural images produces many zerotrees, especially at higher thresholds. Using this concept of the zerotree, the EZW algorithm is based on the zerotree hypothesis.

#### 4.3.1.1 The Algorithm

The EZW encoder can be broken up into five steps which are detailed below.

a) Initialization – Choose the initial threshold $T_h = T_{h0}$ such that $|W_T(x,y)|<T_{h0}$ and at least one transform value satisfies $|W_T(x,y)|>(T_{h0}/2)$. Normally, $n_0 = \lfloor \log_2(max(|W_T(x,y)|) \rfloor$ and $T_{h0} = 2^{n_0}$, where $W_T(x,y)$ are the wavelet transform coefficients and *max()* indicates the maximum coefficient value in the transform. A "header" is output, which consists of the transform size as well as a representation of the initial threshold, $n_0$.

b) Dominant Pass – Each coefficient in the transform is compared to the threshold $T_h$ in a specific order, to test its significance. The most commonly used scanning order is the raster scan which is shown in Figure 4-14. A coefficient is significant if $|W_T(x,y)|\geq T_h$. If the coefficient is significant then it is encoded as 'POS' (positive) or 'NEG' (negative) depending on its sign. This coefficient does not need to be coded again at lower threshold levels; therefore its value in the transform will be set to zero. If the coefficient itself is insignificant but one of its descendants is significant, it is encoded as 'IZ' (isolated zero). If the zerotree hypothesis is true for the coefficient, it will be encoded as 'ZTR' (zerotree root). If the coefficient is coded as a zerotree root, all of its descendants don't need to be encoded as they will be reconstructed as zero at this threshold level. At the end of the dominant pass, all the coefficients that are in absolute value larger than the current threshold are extracted and appended without their sign on the subordinate list and their positions in the image are filled with zeroes.

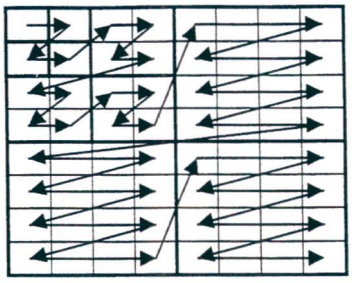


Figure 4-14: Raster Scan Order

c) Subordinate pass – This is the refinement pass (refines the significant coefficient value) in which the next most significant bit of all the coefficients in the subordinate list are output. Each value in the subordinate list is compared to the current threshold $T_h$. If the value is larger than $T_h$, a '1' is output and the current threshold is subtracted from the coefficient value in the subordinate list. If the coefficient is smaller than the threshold, a '0' is output. On completion of the subordinate pass, the subordinate list should be sorted in order of highest to lowest values. This is done so that the larger coefficients, which carry the most information, are in the front, and are thus coded first. Also, the entropy encoder, which follows this quantization stage, will become more efficient, since this pass will generate a group of '1' symbols followed by a group of '0' symbols as apposed to randomly generating these symbols.

d) Decrease the threshold – $T_h = T_h / 2$

e) Repeat - If the threshold is greater than the minimum threshold then repeat Steps (b) to (d). The minimum threshold value controls the encoding performance and also the bit-rate. If a "0" minimum value is specified, it gives rise to lossless reconstruction of the image.

The decoding algorithm is similar to the encoder. The size of the reconstructed transform as well as the initial threshold can be acquired from the "header". The reconstructed transform will initially contain all zero values. The decoder also consists of dominant and subordinate passes. It is important to use the same scanning order that was used by the encoder.

#### 4.3.1.2 Performance and Efficiency

EZW algorithm has very good performance in terms of the PSNR (Table 4-2) in low bit-rate comparison with existing encoders at that time (such as the JPEG standard - Table 4-3), because it preserves all significant coefficients at each scale. The scan order is such that low frequency information, which is dominant in natural images, is output first. Also high frequency

information, caused by edges or object boundaries is also accounted for, since they produce significant coefficients at the high frequency subbands.

However, the drawback of the EZW algorithm is that it is computationally expensive. Also encoding of sub-images is not possible since the entire image must be transformed before encoding begins. To reduce the computation time, Shapiro introduced a more efficient method of identifying zerotrees during the encoding stage [47]. This new algorithm utilizes lookup tables instead of the original recursive search algorithm.

| | Lena | Barbara |
|---|---|---|
| **Bit-rate (bpp)** | **PSNR (dB)** | **PSNR (dB)** |
| 0.25 | 33.17 | 26.77 |
| 0.5 | 36.28 | 30.53 |
| 1 | 39.55 | 35.14 |

Table 4-2: PSNR results for EZW [46] [78]

| | Lena | Barbara |
|---|---|---|
| **Bit-rate (bpp)** | **PSNR (dB)** | **PSNR (dB)** |
| 0.25 | 31.67 | 25.10 |
| 0.5 | 34.90 | 28.49 |
| 1 | 37.94 | 33.26 |

Table 4-3: PSNR results for JPEG compression standard [79]

### 4.3.2 Set Partitioning In Hierarchical Trees

A number of wavelet coding methods, using the fundamental ideas of the EZW coding scheme, have been proposed. One of the most popular algorithms is the SPIHT algorithm introduced by Said and Pearlman [48]. SPIHT was able to achieve higher performance than the EZW algorithm without having to use an entropy encoder. Hence the complexity reduction was significant. This algorithm also formed the basis for future wavelet based coders. Therefore this section is dedicated in detailing the important properties of this algorithm.

#### 4.3.2.1 Principles

The term hierarchical trees refer to the quadtrees while set partitioning refers to the way these quadtrees partition the wavelet transform values at a given threshold. The SPIHT algorithm uses a partitioning of the quadtrees in a manner that tends to keep insignificant coefficients together

in larger subsets. The partitioning decisions are binary decisions that are transmitted to the decoder. SPIHT is a fully embedded wavelet coding algorithm that progressively refines the most significant coefficients. Hence the ordering data is not explicitly transmitted.

#### 4.3.2.2 The Algorithm

The following sets of coordinates are used in the SPIHT algorithm:

- H: set of all coordinates of all spatial orientation tree roots (nodes in the highest pyramid level, the lowest resolution)
- O(i,j) set of coordinates of all offspring's of node (i,j)
- D(i,j): set of coordinates of all descendants of node (i,j)
- L(i,j): set of coordinates of all the descendants of the coefficients at location (i,j) except for the immediate offspring's of the coefficient at location (i,j), i.e. L(i,j)=D(i,j)-O(i,j).

Three lists are then used to keep track of the order in which elements are tested for significance. These lists are:

- LSP – The list of significant pixels which contains the coordinates of coefficients that are found to be significant.
- LIP – The list of insignificant pixels which contains the coordinates of coefficients that are insignificant.
- LIS – The list of insignificant sets, which contain the coordinates of the roots of sets of type D or L.

A set or pixel is significant if the absolute of the maximum element value in that set is greater than the current threshold (Equation (4.23)). The following steps implements the SPIHT encoder:

a) Initialization – Choose the initial threshold $T_h$ (same as for the EZW (Section 4.3.1.1)). Set LIP = H, set LSP = 0 and LIS = D. Output "header".

b) Sorting Pass –

(i) Examination of LIP:

- If the coefficient at the coordinate is significant then output a '1', followed by a bit representing the sign of the coefficient (assume '1' for positive, '0' for negative). Move that coefficient to the LSP list.
- If the coefficient at that coordinate is not significant, output '0' and keep it in the LIP.

(ii) Examination of LIS

- If the set of co-ordinate $(i,j)$ is not significant, then output a '0'
- If the set of co-ordinate $(i,j)$ is significant, then output a '1'. What happens next depends on if the set is of type D or L.
- If it is type D, we check each offspring of the coefficient at that co-ordinate. If the offspring coefficient is significant then output a '1', followed by a bit representing the sign of the coefficient ('1' for positive, '0' for negative). Thereafter move that coefficient to the LSP. If the coefficient at that coordinate is not significant, output a '0' and add their coordinate to the LIP.
- If the set is of type L, add it to the end of the LIS as the root of a set of type D. Note that these new entries in the LIS will be examined during this pass. Thereafter remove the co-ordinate $(i,j)$ from the LIS.

c) Refinement Pass – Examine the coefficients in the LSP and output the $n^{th}$ most significant bit of that coefficient.

d) Decrease the threshold – $T_h = T_h / 2$

e) Repeat – If the threshold is greater than the minimum threshold then repeat steps (b) to (d).

#### 4.3.2.3 Performance and Efficiency

The results in Table 4-4 show that even with the less extensive processing, the SPIHT method outperforms the EZW algorithm in terms of compression and recovered image quality. This is because the SPIHT algorithm does not scan coefficients in a predetermined order like the EZW, which uses the raster scan. Thus the SPIHT algorithm outputs and encodes significant descendants immediately. The SPIHT algorithm is fully embedded and an exact bit-rate can be achieved.

| | **Lena** | **Barbara** |
|---|---|---|
| **Bit-rate (bpp)** | **PSNR (dB)** | **PSNR (dB)** |
| 0.25 | 34.13 | 27.58 |
| 0.5 | 37.24 | 31.39 |
| 1 | 40.45 | 36.41 |

Table 4-4: PSNR results for SPIHT [52] [78]

### 4.3.3 Stack-Run Image Coding

Stack run (SR) coding, developed by Tsai et al [50], is based on the principles of run-length coding. Although this coding scheme is not limited to the wavelet transform, it relies heavily on the fact that the transform quantizes many coefficients to zero. These quantized coefficients can then be partitioned into two groups, namely zero-valued and nonzero-valued (or significant) coefficients. This coding method breaks away from the zerotree quantization concept.

#### 4.3.3.1 The Algorithm

Like in the run-length coding method [14], the SR scheme represents coefficients in the form of $(a,b)$, where $a$ is the number of zero-valued coefficients before the next significant coefficient, and $b$ is the magnitude and sign of that significant coefficient. Four symbols are developed for this algorithm, namely '0' and '1', for the respective binary 0 and 1 of the significant coefficient (i.e. for type $b$), and '-' and '+' for the respective binary 0 and 1 for the coefficient of type $a$. Finally, the sets $\{a\}$ and $\{b\}$ are coded independently using the adaptive arithmetic coder, with separate probability tables to take advantage of the different statistical nature of each set.

A gain in the performance of the algorithm can be achieved by not coding the most significant bit (MSB) of the binary run-length and significant coefficient values. These values, as well as the signs of the significant coefficients, are implied by the structure of the symbol stream. Further reduction in bit-rate can be achieved by quantizing all coefficients below a certain threshold to zero, especially in the finer scale subbands of the transform.

#### 4.3.3.2 Performance and Efficiency

The SR algorithm is conceptually very simple and has extremely low computational complexity. Giving its low complexity, the coding performance (Table 4-5) is consistently better than the basic EZW algorithm. This scheme shows competitive performance with other wavelet based image coders, with a slightly lower performance. However, experimental results show that the SR coding scheme has an increased computational time by about 13 % over SPIHT.

| | Lena | Barbara |
|---|---|---|
| **Bit-rate (bpp)** | **PSNR (dB)** | **PSNR (dB)** |
| 0.25 | 33.63 | 27.39 |
| 0.5 | 36.79 | 30.98 |

Table 4-5: PSNR results for Stack-Run [50]

### 4.3.4 Embedded Conditional Entropy Coding of Wavelet Coefficients

Xiaolin Wu [52] suggested that adaptive context modelling and conditional entropy coding of wavelet coefficients is more important than the type of wavelet transform or the coefficient quantization, in terms of coding efficiency. Hence he developed a coding structure based on a higher order context modeller that is more adaptive to the statistics of the wavelet coefficients. He named his scheme the embedded conditional entropy coding of wavelet coefficients (ECECOW). He noticed that the zerotree structure serves as a high-order context model of small wavelet coefficients. However, this structure poses a large artificial structure on the wavelet coefficients, and does exploit other statistical dependencies of wavelet coefficients that do exist. In other words, the zerotree structure only captures contexts of a square shape in the spatial domain, whereas statistical dependent wavelet coefficients may form regions of arbitrary shapes. Hence the goal of the ECECOW method is to adaptively capture these spatial varying structures.

#### 4.3.4.1 The Algorithm

The ECECOW algorithm only concerns itself with the entropy coding of the quantized wavelet coefficients. The wavelet transform is taken as given, while the quantizer uniformly quantizes the wavelet coefficients and applies bit-plane coding. Hence, all quantized wavelet coefficients are represented as sequence of binary symbols. This sequence is then adaptively binary arithmetically coded, using the simplest and fastest version of the arithmetic coder.

The ECECOW algorithm provides a good estimate of a coefficient so that it can be coded using the adaptive binary arithmetic coder. Each coefficient's probability is conditioned on previous coefficients. This process is called modelling context and it is this that determines the bit-rates in compression algorithms. The aim of the ECECOW scheme is to estimate the conditional probability of a wavelet coefficient based on past coded bits and to use this estimate to drive an adaptive binary arithmetic coder.

It has been noted that wavelet coefficients of similar magnitude statically cluster in frequency subbands and in spatial locations. Also, large wavelet coefficients in different subbands tend to be constructed from the same spatial locations. ECECOW takes advantage of these observations and models a coefficient by its neighbours in the same subband, and by its parents in higher level subbands derived from the same spatial location. The modelling also accounts for structures and features of the different subbands. For example in the HL subband, vertical structures are dominant; hence the modelling considers coefficients that are vertically neighbours of that coefficient. The same logic is applied to the LH and HH subbands.

After the above adaptive context selection (based on subband orientations) is made, the selected modelling event is quantized. A minimum entropy quantizer is used. The signs of the coefficients are also coded based on an estimated probability, which exploits neighbouring coefficients and parents.

#### 4.3.4.2 Performance and Efficiency

ECECOW is competitive against all other wavelet based image compression methods, showing improved results in terms of bit-rates and recovered image quality. The performance of this algorithm is mainly due to the higher order adaptive context modelling. Also, ECECOW produces an embedded coding scheme. The computational complexity of this algorithm is less as compared to other wavelet encoding algorithms, showing a 20% speed improvement over the SPIHT method with a superior R-D performance. Due to the coders' high performance, improvements have been made to further enhance the performance and efficiency. Table 4-6 details the gain of the new ECECOW encoder over the old.

| | ECECOW | | ECECOW New | |
|---|---|---|---|---|
| | Lena | Barbara | Lena | Barbara |
| Bit-rate (bpp) | PSNR (dB) | PSNR (dB) | PSNR (dB) | PSNR (dB) |
| 0.25 | 34.81 | 28.85 | 34.89 | 29.21 |
| 0.5 | 37.92 | 32.69 | 38.02 | 33.06 |
| 1 | 40.85 | 37.65 | 41.01 | 38.05 |

Table 4-6: PSNR results for old and new versions of ECECOW [52] [53]

### 4.3.5 Space Frequency Quantization (SFQ)

Xiong et al [51] presented an image compression scheme that focuses on optimising the joint use of spatial quantization modes and scalar quantization. This coder exploits both the frequency and spatial compaction property of the wavelet transform, by means of the two quantization modes. Spatial quantization is achieved via the zerotree quantization method while the rest of the coefficients which have not yet been coded, using the zerotree process, are uniformly scalar quantized independent of the coefficient's frequency band. Hence they named their coder space frequency quantization (SFQ).

Zerotree quantization is based on the assumption that a spatial region of high-frequency coefficients has a zero value. In other words, most of the energy of a typical image is concentrated in the lower frequency subbands and high frequency components corresponding to energy that is spatially concentrated around edges. In the SPIHT and EZW algorithm, zerotree quantization fails when there is a significant coefficient present in the higher frequency subbands. Hence theses algorithms are not optimized completely in terms of zerotree quantization. The SFQ methods seeks to redefine the zerotree quantization, in that the quantization is performed based on a rate-distortion criterion, rather than a significance factor.

#### 4.3.5.1 Algorithm

The SFQ coder aims to optimize the application of zerotree and scalar quantization in order to minimize the distortion for a given rate constraint. The algorithm focuses on optimally selecting spatial regions for applying the zerotree quantization, and searches for an optimal scalar quantizer step-size for quantizing the remaining coefficients. It has been motivated in [51] that a single step-size should be used in the scalar quantizer. One of the reasons for this is that the zerotree quantization process removes the bulk of the insignificant coefficients, thus removing the "peakiness" in the distribution of subbands and hence leaving these subbands with near-flat distributions.

The algorithm consists of two parts, with the first being the decision of how to divide a tree in the wavelet transform, such that zerotree quantization can be applied to it. The approach taken is to start at the second highest frequency subbands and for every node in the given subband, determine whether it is cheaper to zero out or keep its descendants in a rate-distortion sense. This approach is then continued to every other coarser scale subband, until the coarsest scale (highest level of the wavelet transform) is reached. The above process is then iteratively applied until an optimal quantization step-size for a given zerotree quantization is established.

The second part of the algorithm is to construct a zerotree map to indicate which coefficients of the tree have been zerotree quantized, and which are not. All surviving coefficients are scalar quantized with the determined optimal quantization step-size. Thereafter, an adaptive arithmetic coder is used to entropy code the quantized wavelet coefficients.

#### 4.3.5.2 Performance and Efficiency

Due to the optimisation of zerotree quantization, the SFQ algorithm is shown to outperform the other zerotree coders, namely SPIHT and EZW. Also, SFQ illustrates competitive results (Table 4-7) with other wavelet based image coders. Due to its iterative zerotree quantization stage, the computational complexity of this algorithm is enormous. Results have shown a 500% increase in the encoding time of SFQ as compared with the SPIHT coder on the "Lena" image. However, the decoding time of SFQ is much faster. Due to the complexity of the algorithm, the authors stated they are not considering adding further improvements to this coder.

| | **Lena** | **Barbara** |
|---|---|---|
| **Bit-rate (bpp)** | **PSNR (dB)** | **PSNR (dB)** |
| 0.25 | 34.33 | 28.29 |
| 0.5 | 37.36 | 32.15 |
| 1 | 40.52 | 37.03 |

Table 4-7: PSNR results for SFQ [51]

### 4.3.6 Comparison of Wavelet Coders

Figure 4-15 and Figure 4-16 compares the R-D performance of the different wavelet image coders on the two images, "Lena" and "Barbara" respectively. These results indicate that the ECECOW algorithm produces the best R-D performance. Furthermore ECECOW has the lowest computational complexity. Another interesting observation is that the wavelet based JPEG-2000 image compression standard has substantial results improvement over the DCT based JPEG standard. The SPIHT method, which also has low encoding times, has comparable outcomes to ECECOW and SFQ algorithms.

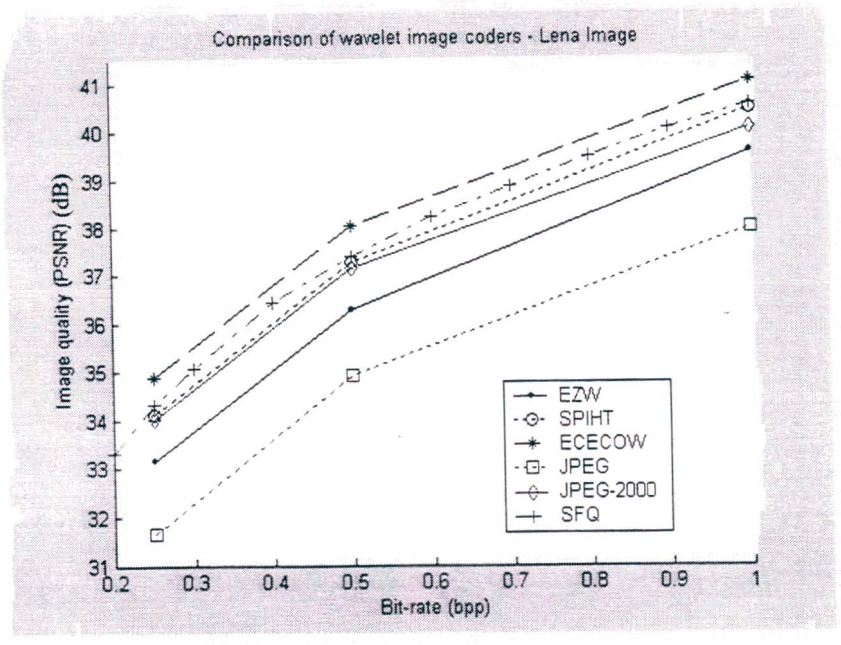



Figure 4-15: Performance comparison between different wavelet image coders and JPEG standards on the "Lena" 512x512 image

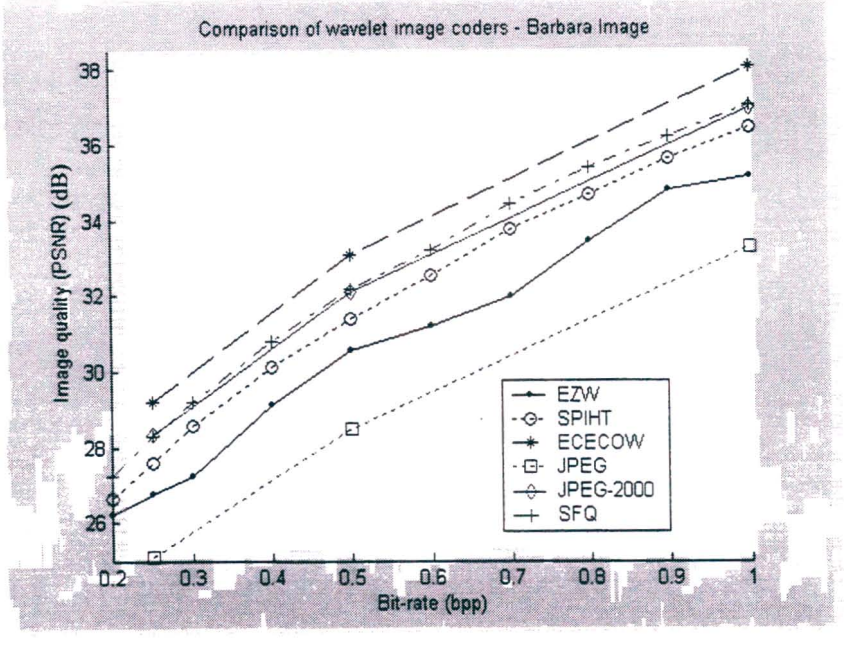



Figure 4-16: Performance comparison between different wavelet image coders and JPEG standards on the "Barbara" 512x512 image

## 4.4 Summary

This chapter has introduced the basic concepts of wavelets and the wavelet transform. In essence, the wavelet transform provides a multi-resolution, multi-frequency representation of an image. Several properties of the wavelet transform of images were discussed, which motivated their role in image compression.

Wavelet compression algorithms are mainly based on the transform coder. It has been shown that changing the wavelet basis to perform the transform has very little effect on the performance of the coder. Mallat's fast wavelet transform method is widely used for computation of the wavelet transform.

Several wavelet based image compression algorithms were investigated. These algorithms differ at the quantization and entropy coding stages. The wavelet image coders have shown performance greater than the JPEG image compression standard. For each algorithm, the performance and complexity was examined. A comparison of the performance showed that the ECECOW and SPIHT coding schemes are the best wavelet coders in terms of R-D performance and low computational complexity.

# CHAPTER 5 - FRACTAL COMPRESSION IN THE WAVELET DOMAIN

In order for fractal block coders to be effective, images must be composed of features at fine scales that are also present at coarser scales i.e. the self-transformability assumption. Another downfall of the fractal coding method is the tiling effect (Figure 3-11), where the artefact edges between range block boundaries appear (especially at low bit-rates). A solution to the tiling effect was to allow range blocks to overlap. However, this scheme reduces the compression ratio, since the same part of the image is coded more than once.

The most likely method of eliminating the tiling effect is to apply the fractal methodology to a transformed representation of the image. This involves filtering the contents of each range block, such that the data of the filtered range block is orthogonal to all neighbouring overlapping blocks. By doing this, neighbouring blocks contain independent information and hence can be coded separately and without redundancy. The obvious choice was to use the wavelet transform since the filtering process is natural and self-similarities do exists in the multi-resolution wavelet representation.

The first significant paper linking fractal and wavelet compression was presented by Rinaldo and Calvagno [26]. They introduced a method that combines fractal and wavelet coding ideas for a better compression method. Their coder, called the predictive pyramid coder (PPC), exploited inter-scale redundancies in the wavelet transform. The PPC coder performs a block-based prediction that predicts finer scale wavelet coefficients from coarser scales. However, this coder vaguely represented the contractive mappings defined by the Jacquin type fractal coder. The link between fractal and wavelet based coding was discovered independently by the workings of G.M. Davis [28], H. Krupnik et al [27] and A. van de Walle [31]. They all generalized fractal coding from the spatial domain to the wavelet domain, and developed the idea of, which Davis termed, fractal wavelet subtree coding.

Section 5.1 extends the theoretical foundations of fractal image compression from the spatial domain to the wavelet domain. The next section (Section 5.2) uses the fundamentals of the previous section to provide a general compression algorithm for fractal coding in the wavelet domain. Thereafter, a literary survey is presented in Section 5.3 and Section 5.4, which compares and assesses currently established wavelet based fractal image and video coders respectively.

## 5.1 Wavelet Analysis of Fractal Block Coding

The wavelet transform is a natural tool for analyzing fractal block coders, since the wavelet basis possess the same type of dyadic self-similarity that fractal coders seek to exploit. In other words, fractal block coders make use of the self-similarity property present across scales of the signal, while the wavelet transform provides an efficient multi-resolution representation of the signal. We will now describe the effect a fractal block coding scheme exhibits in the wavelet domain.

### 5.1.1 Spatial Domain

For simplicity, the discussion is reduced to a one dimensional signal and the idea is later extended to two dimensions, for images. Suppose we are given a signal vector $f(x)$ and wish to perform a fractal compression on it. Using the Jacquin style algorithm (Section 3.2.4), $f(x)$ is partitioned into non-overlapping range vectors, $R_n$, of size B and domain vectors $D_p$, of size 2B. The first step is to average and sub-sample the domain vector $D_p$ to size B i.e. obtaining the vector $\hat{D}_p$, as in Equation (3.33), but for a one dimensional signal (averaging pairs of adjacent samples).

Instead of averaging and sub-sampling each domain block individually, we could average and sub-sample the entire signal first to obtain:

$$f_{1/2}(x) = Av/s(f(x)), \tag{5.1}$$

where $Av/s$ is the average and sub-sample operator. We can then choose the averaged and sub-sampled domain vector $\hat{D}_p$ from $f_{1/2}(x)$.

Continuing with the fractal coder, each range vector will be approximated by a transformed domain vector according to Equation (3.34). Hence, the approximation for the range vector $R_n$ is given by

$$R_n \approx s_i T_j(\hat{D}_p - \bar{d}_p) + \bar{r}_n, \tag{5.2}$$

where, $s_i$ is the scaling factor and $T_j$ an affine transformation. For simplicity, again let us assume that the transformation only consists of the scaling factor, hence Equation (5.2) reduces to:

$$R_n \approx s_i(\hat{D}_p - \bar{d}_p) + \bar{r}_n. \tag{5.3}$$

### 5.1.2 Wavelet Domain

Let us now analyse the effect the mapping Equation (5.3) has in the wavelet representation of the signal. The Mallat algorithm (Figure 4-5) is used in the decomposition of the wavelet transform, with the wavelet basis being the Haar wavelet. The Haar wavelet transform is an orthonormal transform that has the effect of averaging the sample values. The lowpass and highpass filter of the Haar wavelet is given by:

$$\begin{aligned} H_L &= [1, \quad 1] \\ H_H &= [1, -1] \end{aligned} \qquad (5.4)$$

The three level decomposition of the function $f(x)$ is shown in Figure 5-1. The output of the higher branch represents high pass filtering and the 2:1 decimation and labelled $f_{H\,1/2^k}$, while that of the lower branch yields the low pass filtering, also with a 2:1 decimation, labelled $f_{L\,1/2^k}$.

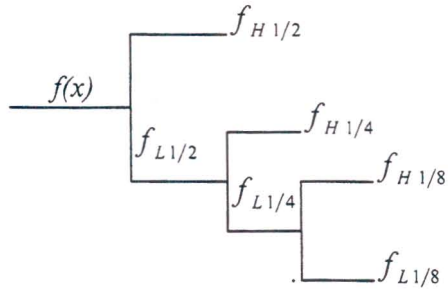



Figure 5-1: Three level DWT representation of the a signal *f(x)* using the Haar wavelet

The 2:1 decimation of the output of the convolution of $f(x)$ and the $H_L$ (for the Haar wavelet) results in $f_{1/2}(x) = Av/s(f(x))$, the averaging and sub-sampling operation. Hence, for the Haar wavelet,

$$f_{L1/2^k} = f_{1/2^k}. \qquad (5.5)$$

We can now decompose both the domain and range vectors using the Haar wavelet transform, with $\log_2(2B)$ splits. This is depicted in Figure 5-2, where B=4 (this implies three levels of the wavelet transform). The range vector will have coefficients $\{r_1, r_2, \ldots, r_B\}$, while the domain vector, coefficients $\{d_1, d_2, \ldots, d_{2B}\}$. Since the Haar wavelet transform is used, we can construct range blocks $R_n$ from averaged and sub-sampled domain blocks, $\hat{D}_p$, which resides in $f_{1/2}$. The wavelet coefficients of $\hat{D}$ are $\{\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_B\}$. Coefficients $r_1$, $d_1$, and $\hat{d}_1$ are the averages

(up to a gain factor) of the vectors $R$, $D$ and $\hat{D}$ respectively. It can also be noted that the set of coefficients $\{d_1, d_2, \ldots, d_B\}$ and $\{\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_B\}$ are equal for the Haar wavelet transform.

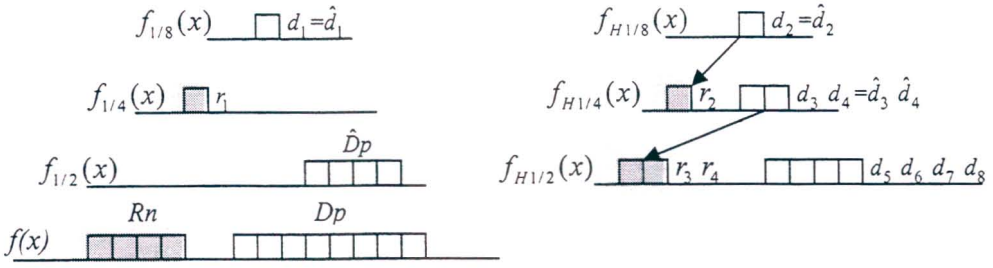



Figure 5-2: Relations between the Haar DWT coefficients between range and domain vectors
Left - lowpass pyramid; Right - highpass pyramid

Applying the spatial domain contraction mapping Equation (5.3) to the wavelet domain results in:

$$\begin{aligned} R_n - \overline{r}_n &= s_i(\hat{D}_p - \overline{d}_p) \quad \rightarrow \text{ Spatial domain} \\ r_q &= s_i.d_q \quad \text{for} \quad 2 \le q \le B \quad \rightarrow \text{ Wavelet Domain} \end{aligned} \tag{5.6}$$

Referring back to Figure 5-2, it can be seen that Haar coefficients for the range vector $\{r_2, \ldots, r_B\}$ and domain vector $\{d_2, \ldots, d_B\}$ are completely contained in the wavelet transform. It is also noted that the domain vector coefficient is at a coarser scale in the wavelet transform than the corresponding range vector coefficient. Hence Equation (5.6) can be interpreted as a prediction of higher frequency subbands from lower ones. For fractal compression in the wavelet domain, the coarsest scale domain coefficients $d_1$ and $d_2$ are stored; together with the scaling factor $s_i$. In decoding, the coarsest scale domain vector coefficient $d_2$, is used to construct the range vector coefficient $r_2$ at the next coarser scale. Continuing with this, for all domain coefficients the next coarser subband is completely produced. Thereafter, at each stage another subband is extrapolated so that a higher resolution of the fixed point can be obtained. This process can be visualized in Figure 5-3. In this figure, during the coding stage, the approximation and detail coefficients at the coarsest scale (level 3, shown as shaded) are stored. A domain vector $D_p$ is extracted from the transform, averaged and reduced in size to produce $\hat{D}_p$. $R_n$ is then constructed using Equation (5.6), with a scaling factor $s_i$.

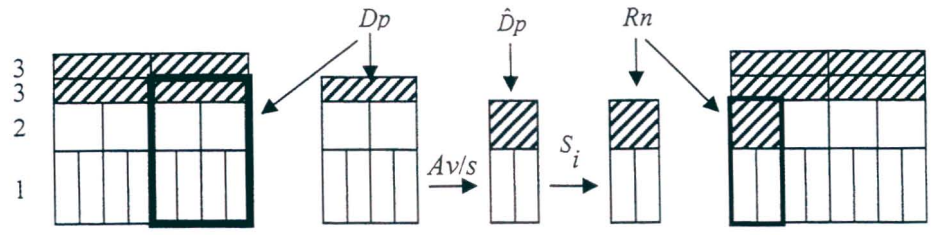



Figure 5-3: Construction of a DWT range vector, given the DWT domain vector and transformation

### 5.1.3 Extension of Analysis to Images

The idea in Section 5.1.2 can easily be extended to two dimensions. The separable 2-D Haar wavelets will be used to compute the wavelet transform of an image. Davis [28] introduced a new term called the wavelet subtree, which consists of all wavelet coefficients from the same spatial location, but with different resolutions and orientations. The wavelet transform of an image block, produces a wavelet subtree together with its associated scaling function coefficient. Figure 5-4 illustrates the three level wavelet decomposition of a range block of size BxB and domain block of size 2Bx2B. In this figure, B = 4. For example, the domain subtree is defined as the low frequency wavelet coefficients at level 3, from all orientations, their children and each of their children all the way to level 1 of the transform (shown as the unshaded blocks). Also depicted in the figure is the range subtree (shaded blocks), which starts at level 2 of the transform.

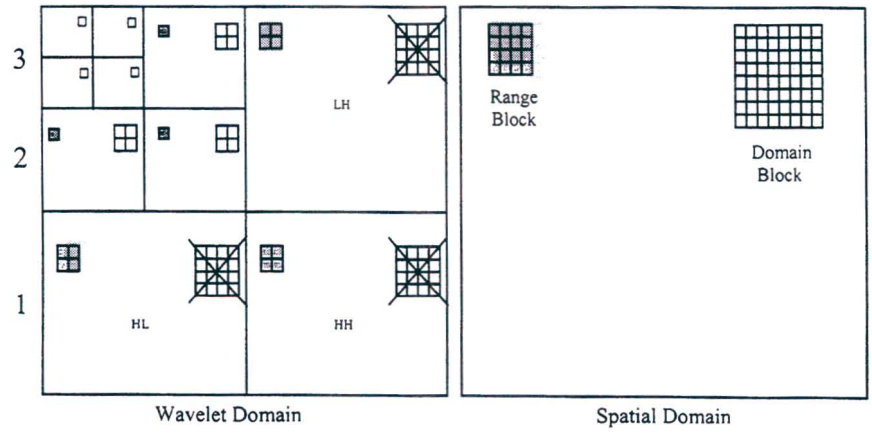



Figure 5-4: Three level wavelet transform of range and domain blocks

As in the one dimensional case, fractal compression in the wavelet domain implies that the domain wavelet coefficients from the coarser scale must be used to predict the range coefficients at the next lower level of the transform. For the two dimensional case, the conversions to the wavelet domain are given below.

a) Averaging and sub-sampling of the domain block – Local averaging of the spatial domain block preserves all Haar wavelet coefficients except for the finest scale ones, which are set to zero. Sub-sampling the spatial block involves obtaining the Haar wavelet coefficients at the next finest scale multiplied by half, i.e. it removes all the finest scale wavelet coefficients. This event is shown in Figure 5-4, where the finest scale wavelet coefficients of the domain subtree are crossed out.

b) Block Transformations or Isometries – From Figure 5-4 it is noted that wavelet coefficients of higher subbands (range blocks) must be predicted from coefficients at the next lower subband (domain block). Table 5-1 details the conversion of the spatial isometry operations to wavelet domain isometries. The same isometry is applied to each of the subbands LH, HL, HH. Note that for some isometries, a switch between the HL and LH subbands occur. Also sign inversions of the coefficients may arise.

c) DC component – In the normal fractal compression scheme, each range block is approximated by a transformed domain block plus a constant block. This constant block is to account for the DC component. Wavelet subtrees have no DC component, so the constant block is not required.

| **Isometry** | **Wavelet Isometry** | | |
|---|---|---|---|
| | Sub-block isometry | Exchange sub-blocks LH and HL | Sign change for sub-blocks |
| Identical | Identical | No | None |
| Horizontal flip | Horizontal flip | No | LH, HH |
| Vertical flip | Vertical flip | No | HL, HH |
| Transpose | Transpose | Yes | None |
| Diagonal flip | Diagonal flip | Yes | LH, HL |
| 90° rotation | 90° rotation | Yes | LH, HH |
| 180° rotation | 180° rotation | No | LH, HL |
| 270° rotation | 270° rotation | Yes | HL, HH |

Table 5-1: Correspondence of the isometry operator in the spatial and wavelet domains [33]

## 5.2 Fractal Compression in the Wavelet Domain

The wavelet analogy of fractal image compression in the spatial domain is to approximate each range subtree by transformed domain subtree. This algorithm is depicted in Algorithm 5-1. In decoding, we decode all wavelet coefficients from the coarsest scale. Thereafter, the nodes of the coarsest scale domain subtrees are used to construct the roots of the range subtree at the next coarser scale, by applying the corresponding scaling factors and isometries. Once all the roots of the range subtree are constructed, the next subband of the wavelet transform is computed. This process continues, until all the coefficients of the finer scale subbands are constructed (Algorithm 5-2). It must be noted that this prediction structure removes the iterative decoding that is characteristic of spatial domain fractal coders.

**Algorithm 5-1: Fractal-Wavelet Subtree Encoder**:

1. Compute DWT – Choose a range block of size B. The number of levels computed in the DWT is $\log_2(2B)$.
2. Extract range and domain subtrees of $B^2$-1 coefficients.
3. For each range subtree find the best matching domain subtree, scaling factor and transformation. The match is carried out over all resolution subbands simultaneously, with a range subtree at level $j$ and a domain subtree of the same size at the next higher level $j+1$.
4. Store the coefficients of the highest level in the wavelet transform i.e. the coefficients from the coarsest scale. Also store the scaling factors, the isometry used, as well as the indices of the roots of the domain subtree used in the prediction.

**Algorithm 5-2: Fractal-Wavelet Subtree Decoder**:

1. Copy the highest level wavelet coefficients from the code stream into the wavelet transform.
2. Extrapolate downwards in the wavelet pyramid – By using the highest level coefficients, the associated scaling factor and isometry, find the coefficients at the next level of the transform. Continue with this all the way down to the first level.
3. Compute the inverse DWT.

### 5.2.1 Blockless IFS

The main benefit gained by performing fractal compression in the wavelet domain is the removal of the tiling (or blocking) effect that are present in fractal coded images. This is achieved by the use of a smooth wavelet basis instead of the Haar wavelet. Smooth wavelet basis eliminates the sharp discontinuities at the range block boundaries caused by quantization errors. However, by changing the wavelet basis, the scaling function is no longer the four pixel averager. The scaling function is now a 2-D separable filter followed by a decimator. The

wavelet representation, using these smooth filters, will cause blocks to overlap in the spatial domain and hence the subtrees do not correspond to blocks in the spatial domain anymore. Therefore employing the compression algorithm (Algorithm 5-1), using a smooth wavelet basis in the wavelet transformation, will have a smoothing effect on the reconstructed images and thus removing the blocking or tiling effect.

### 5.2.2 Performance and Efficiency

The performance of the fractal wavelet subtree coder (Algorithm 5-1) was evaluated by means of simulation on the "Lena" 256x256 image. Once again the "Matlab" simulation engine was used. Table 5-2 details all the results obtained. Two types of wavelet basis, namely the Haar wavelet and Biorthogonal 4.4 wavelet, were used in the experiments. Different fixed block sizes were used in the fractal compression, which corresponded to certain levels of the wavelet transform for the subtree coders. Figure 5-5 and Figure 5-6 depicts the recovered images for the FBC and subtree coders, coded with spatial domain range block sizes of 8x8 (corresponding to 4 levels of the wavelet transform) and 16x16 (corresponding to 5 levels of the wavelet transform) respectively.

The results illustrate that fractal coding in the wavelet domain (subtree coders) achieves higher performance, in terms of image compression and recovered image quality, than the standard fractal block coders. Also, there is a dramatic reduction in the compression and decompression times.

From Figure 5-5 and Figure 5-6, it is noticed that using the Haar wavelet produces the similar tiling structure present in normal fractal coders. However, using other wavelet families such as the Biorthogonal family in this case, have the effect of smoothing out the image. This consequently removes the tiling structure and hence improving the image quality. The other trend that is noticed is that as the block sizes increases or the number of levels of the wavelet transform increases, the compression increases, but the image clarity decreases. This is to be expected, since fewer blocks (or trees) now cover the image. Another observation is that the coding and decoding times reduce.

| Fractal Block Coder | | | | |
|---|---|---|---|---|
| **Range Block Size** | **Compression Ratio** | **PSNR (dB)** | **Encoding Time (s)** | **Decoding Time (s)** |
| 8x8 | 18.2 | 25.48 | 655.3 | 9.89 |
| 16x16 | 81.9 | 21.34 | 104.6 | 9.17 |
| 32x32 | 356.2 | 17.82 | 22.6 | 9.01 |
| **Wavelet Domain Fractal Subtree Coder – Haar Wavelet** | | | | |
| **Levels of Wavelet Transform** | **Compression Ratio** | **PSNR (dB)** | **Encoding Time (s)** | **Decoding Time (s)** |
| 4 | 23.27 | 26.17 | 209.42 | 2.29 |
| 5 | 97.52 | 21.40 | 19.48 | 1.06 |
| 6 | 455.11 | 17.61 | 2.62 | 1.03 |
| **Wavelet Domain Fractal Subtree Coder – Bior4.4 Wavelet** | | | | |
| **Levels of Wavelet Transform** | **Compression Ratio** | **PSNR (dB)** | **Encoding Time (s)** | **Decoding Time (s)** |
| 4 | 23.27 | 27.09 | 204.12 | 2.25 |
| 5 | 97.52 | 22.31 | 19.40 | 0.98 |
| 6 | 455.11 | 18.43 | 2.75 | 1.20 |

Table 5-2: Performance comparison between the fractal block coder and the wavelet domain fractal subtree coders on the "Lena" 256x256 image.


Figure 5-5: Comparison of recovered images between the fractal block coder and the wavelet domain fractal subtree coders
(a) Original image, (b) Fractal coded - block size 8x8,
(c) Subtree coded - Haar wavelet (4 level wavelet transform), (d) Subtree coded - Bior4.4 wavelet.

Figure 5-6: Comparison of recovered images between the fractal block coder and the wavelet domain fractal subtree coders
(a) Original image, (b) Fractal coded - block size 16x16,
(c) Subtree coded - Haar wavelet (5 level wavelet transform), (d) Subtree coded - Bior4.4 wavelet.

## 5.3 Combination of Fractal and Wavelet Image Coders

There are a few image compression schemes that have been developed which are based on fractal coding in the wavelet domain. These systems utilize the algorithm outlined in Algorithm 5-1. Each of these compression schemes will be discussed briefly, outlining the important aspects of the algorithm.

### 5.3.1 Predictive Pyramid Coder

The predictive pyramid coder (PPC) was developed by Rinaldo and Calvagno [26], before the link between fractal coding and the wavelet representation was found. This coder aimed at

exploiting similarities among detail signals in the multi-resolution wavelet representation. They noticed that blocks in different subbands with the same orientation have similar bandpass characteristics, which suggests the visual closeness of these sub-bands. The main focus of their algorithm was to predict blocks in a subband from blocks in subbands with the same orientation, but at a lower resolution.

#### 5.3.1.1 Algorithm

Their scheme consisted of two routines, namely the block prediction portion and the residual block coding routine. In the block prediction method, the coefficients of the lowest resolution subbands are quantized with a 7 bit Laplacian quantizer. Starting at the next coarser subband, the entire subband is divided into range blocks. Then, an affine transformed domain block (normally four rotations) from the previous coarser subband (of the same orientation), is sought that matches range blocks in the current subband in terms of the MSE. The range and domain blocks are of the same size, therefore no averaging and sub-sampling is required. If a match is not found for a particular range block, the quadtree approach is taken, in which the range block is reduced in size. If that particular range block, still cannot be coded using the prediction method, then it is coded by the residual block coding scheme. The prediction method continues subband by subband until the finest scale subband transform coefficients are coded. An advantage of this prediction coding scheme is that there is no need to force contractivity, since the prediction of subband sub-images is from lower resolutions ones, whose coefficients are known. Hence the scaling factor can take on any value. Also, the offset factor is not required.

Every range block that has not been coded using the prediction method is coded using the residual block coding routine. Here, each block is quantized using the Laplacian quantizer and encoded with the Huffman coder.

Only one step decoding is required, where each finer scale subband is constructed from the previous coarser subband, together with its prediction parameters.

### 5.3.2 Self Quantization of Subtrees

While analysing the effect that fractal compression has in the wavelet domain, Geoffrey Davis, introduced the self quantization of subtrees (SQS) coding scheme [28]. This coding scheme is the wavelet-based analogue of fractal compression if the Haar wavelet is used. Davis also showed that by using other smooth wavelet basis, the SQS compression method outperforms the best fractal based schemes by about 1 dB in PSNR across a broad range of compression ratios.

#### 5.3.2.1 Algorithm

Davis's algorithm is similar to Algorithm 5-1, with the exception to the number of levels computed by the wavelet transform. If the size of the image is $2^N x 2^N$, then $N$ levels of the transform is calculated. Thereafter coarse scale wavelet coefficients up to scale $N-B-1$ are stored by scalar quantization. These coarse scale coefficients are used to predict the finer ones.

Davis also noted that by using smooth wavelet basis, numerous zero subtree structures are formed. Hence these structures need not be stored by a symmetry operation on a domain subtree, and thus these subtrees can be coded efficiently. One of the most important aspects of Davis's work is the reconstruction theorem, which proves the convergence of SQS reconstruction.

**Theorem 5-1: Reconstruction Theorem**: Let A be a $2^N x 2^N$ image and suppose the coarse scale wavelet coefficients of A up to scale *J-1* are known. Also suppose that each subtree with a root at scale $J$ is constructed from a transformed subtree from scale *J-1*. Then we can reconstruct A in *N-J* steps [28].

The above theorem can intuitively be seen, by noticing that in the decoding process, we start with the coarsest scale coefficients and iterate the map to generate the finer scale coefficients. In other words, the roots of the domain subtrees are used to construct the roots of the range subtrees. By applying the corresponding mappings to each domain subtree root, the entire band of range subtree roots are constructed. Each time we apply the mapping, one more band of coefficients are constructed. Hence the algorithm converges after *N-J* steps.

It is also noted that this map from coarse scale to fine scale allows fractal compression schemes to decode images at higher resolutions than what they were encoded at. This happens since fine-scale coefficients of the high resolution images are extrapolated from the same map used to generate the subtree coefficients.

### 5.3.3 Wavelet Fractal Coder

Li and Kuo introduced the first hybrid wavelet-fractal scheme that, unlike the conventional fractal coder where the whole image is encoded by fractal methods alone, fractal prediction is only applied to selected parts of the image [33]. This algorithm applies the fractal coding method, wherever it is more efficient. The efficiency is evaluated by an adaptive rule that compares the rate saving achieved by the fractal method with the number of bits required by the prediction. Fractal prediction is implemented whenever that rate saving is greater.

#### 5.3.3.1 Algorithm

The first step of the wavelet fractal coder (WFC) is to attempt to fractal code subtrees according to the algorithm specified in Algorithm 5-1. The efficiency of each predicted range subtree is evaluated with respect to the abovementioned R-D criterion. If fractal prediction is chosen, the prediction parameters such as the position of the domain subtree, wavelet isometry and scaling factor are stored. On the other hand, if the matching criterion is not met, then that range subtree is coded using the bit-plane wavelet coder. All coarsest scale wavelet coefficients are also coded using the bit-plane coder. The bit-plane coder starts encoding wavelet coefficients from coarse scale to fine scale.

Experiments have been performed to achieve further reduction in the bit-rate, by varying several parameters in the fractal coding portion of the WFC. Results have shown that fractal prediction is able to reduce the coding rate in certain bit consuming areas, such as texture areas. Also, the performance of the WFC can be improved by further encoding the fractal prediction residue by the bit-plane coder.

### 5.3.4 Variable Tree Size Fractal Coder

A downfall of the compression algorithm stated in Algorithm 5-1, is that the coding process of the fixed size range subtree is completed regardless of the final distortion since only the minimum distortion is sought. Hence, this process incurs large reconstruction error, which leads to very poor image qualities, especially at high compression ratios. As was alluded to earlier (Section 3.2.5) in block based fractal compression, to achieve a better compression algorithm, we must take into consideration the properties of the image. Therefore the quadtree method must be extended to the wavelet domain. Hence, this requires splitting of the wavelet subtrees according to its local details. The abovementioned method was presented by Zhang and Po [35], in which they used a variable tree size partitioning method to obtain a good trade-off between image quality and compression ratio.

#### 5.3.4.1 Algorithm

In their algorithm, for a given range subtree, a best matched transformed domain subtree must be sought as in Algorithm 5-1. If the distortion between the range subtree and transformed domain subtree is less than a predefined threshold, then the range subtree is fractal quantized. The distortion measure is normally the MSE. On the other hand, if the distortion is greater than the threshold, the range subtree is split into four children subtrees. The three root nodes are removed from the tree and scalar quantized. This is done since the coarse scale coefficients

usually possess more energy as compared to the finer scale ones. The length of these subtrees have changed, therefore we need to construct a new domain search pool consisting of nodes at the next level of the transform.

This process is shown in Figure 5-7, where the shaded regions represent the pruned range and domain subtrees. The distortion check is performed again on the pruned subtrees. This adaptive partitioning scheme continues until all the children range subtrees are coded or a minimum dimension range subtree is reached, in which the range subtree is coded with the least distortion.

An improvement to the above algorithm can be accomplished by noticing that at each subtree spilt, the four range subtrees are highly correlated in both the wavelet and spatial domain. Hence these four subtrees can be combined representing a new range subtree. If this combined range subtree can find a well-matched domain subtree (within a predefined distortion threshold), nearly 75 % reduction in bit allocation will be obtained. If a match cannot to found, then only the range subtree is split into four individual subtrees. This scheme by Zhang and Po achieved a slight coding efficiency gain over the SQS method, but the complexity has been significantly reduced.

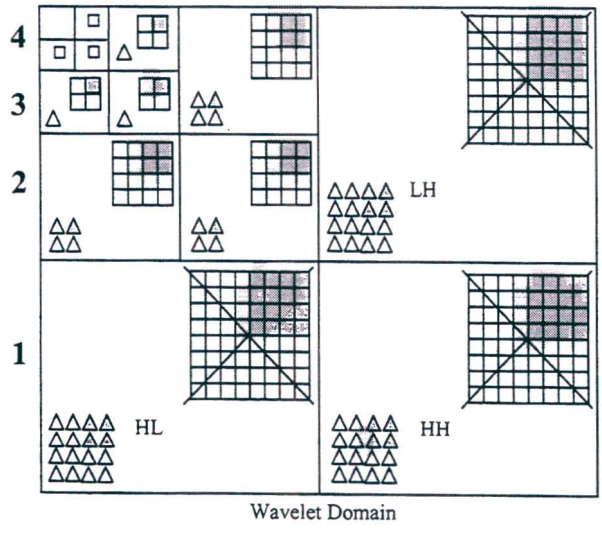



Figure 5-7: Range and domain subtree splitting for the variable tree size fractal coder [35]
Range subtrees are triangles and domain subtrees are squares

### 5.3.5 Fractal Zerotree Wavelet

The fractal zerotree wavelet (FZW) algorithm was developed by Kim et al [36], to seek improvement in the recovered image quality of EZW coders in a local region near the specified bit-rate. The basic idea behind this algorithm is to choose between an EZW (or SPIHT) and a tree-based fractal encoded subtree according to a locally optimized rate-distortion calculation.

This scheme is different from the WFC in that the wavelet compression algorithm is performed first, and fractal compression of subtrees is used merely to enhance the performance of the coder.

It has been noticed that in some cases the use of the explicit zerotree structure in EZW algorithms for a given bit-plane can cost a substantial number of bits with very little contribution to the image quality. One such case is when there are isolated zeros present in the finer scale subbands of subtrees. This area typically corresponds to highly textured areas and edges. Wavelet domain fractal coders, on the other hand, are good at representing regions of constant gradients, textured areas and straight edges that have self-similarity. Thus the wavelet domain fractal method is adaptively applied to parts of the image that can be coded more efficiently than with an EZW coder.

#### 5.3.5.1 Algorithm

This scheme begins with a trial encoding and trial decoding method, in which the EZW (or SPIHT) algorithm is used to compress and decompress the image to a target bit-rate (and slightly higher). The instantaneous slope of the R-D curve developed can be computed near the target bit-rate. This value is used as the distortion measure for which each fractal coded subtree is compared against. Hence a range-domain fractal subtree search is performed, as described in Algorithm 5-1. If the matching criterion is below the defined distortion then the range subtree is fractal encoded and the coded bits used to represent this subtree are removed from the EZW bit-stream.

To retain and enhance the progressive property of the EZW coders, the fractal decoding process is performed at each bit-plane threshold. The increase in computational complexity is not that significant, since only a single iteration in fractal decoding is performed. At each threshold level, the decoded fractal range subtrees are overwritten with a higher resolution one. This is done as to make the FZW even more progressive, and decoding can be performed at any required resolution.

In the FZW scheme, it has been noticed that as more subtrees are fractal coded, the RD gain over EZW increases. However, there comes a point when further increase in fractal coded subtree will cause the gain to shrink. This occurs, since the improvement by fractal coding can only be achieved over a region of rates less than but in the region of the target bit-rate. It has been shown that by basing FZW on the SPIHT coder, it can outperform SPIHT.

### 5.3.6 Performance Comparison

Figure 5-8 compares the R-D performance of the different image coders that combine fractal and wavelet techniques. The results indicate that FZW and WFC achieve the best performance. The performance gain of these coders is due to the majority of the image being coded by the wavelet algorithms. The fractal method is used only in regions of the image where there are constant gradients and textured areas. This is where the performance of fractal coders is extremely high. However, these wavelet domain fractal coders show significant improvement over the fractal block coders.

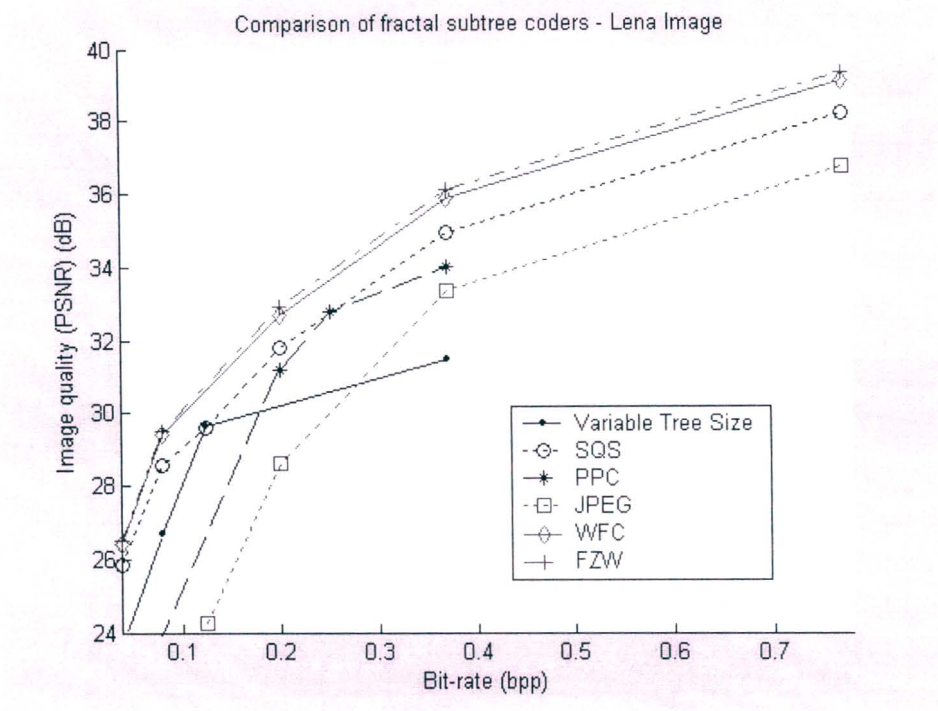



Figure 5-8: Performance comparison between different wavelet domain fractal subtree image coders and JPEG standard on the "Lena" 512x512 image

## 5.4 Combination of Fractal and Wavelet Video Coders

There are few wavelet based fractal video coders that do exist in literature. Two video coding algorithms that utilize these techniques are presented below.

### 5.4.1 Variable Tree Size Fractal Video Coder

Zhang, Po and Yu [37] proposed a video coding algorithm based on their wavelet variable tree size fractal image coder (Section 5.3.4). They proposed and motivated that a subtree structure can characterize motion aptly, in that the subtree representation can describe large object motion

activities. Thus, all the video coding was performed in the wavelet domain. Their coder took the form of the intra/inter frame coder, in which the range subtrees were constructed from coefficients in the current frame, while the domain subtrees were formed in the previous frame. Using this method implied that both the range and domain subtrees were of the same size, and constructed considering the coarsest scale coefficients.

#### 5.4.1.1 Algorithm

The video compression algorithm consisted of two parts, namely the intra-frame and inter-frame coding section. All intra-frame coding was performed using the variable tree size fractal image coder [35]. The first frame of a sequence was always intra coded. A frame memory was used to provide a reference for the inter-frame coding stage. After the compression of each frame, the memory was refreshed by decoding that frame.

The first step of their inter-frame algorithm was to perform a multi-resolution motion detection operation. This operation classifies subtrees into two categories; motion and non-motion. The detection is based on the MSE. If the MSE between a range subtree and its corresponding domain subtree (from the previous frame), is less than a predefined threshold, then that range subtree is a non-motion subtree. Otherwise, it is a motion subtree. The coding of the non-motion subtree is reduced to a single bit.

The motion subtrees are coded by a similar approach taken in their variable tree size fractal image coding algorithm. The general idea is that if a range subtree cannot find a transformed domain subtree, within a given distortion, the range subtree is split into their children subtrees. In other words, the parent nodes are removed and scalar quantized. Each child range subtree then tries to find a matching transformed domain subtree of the same size within a distortion. If there is still no match, the splitting process is repeated. This continues until a minimum range tree size is reached. It is here where that small range subtree is coded with the best-matching domain subtree, even if the approximation is poor.

#### 5.4.1.2 Performance and Efficiency

This video codec has obtained bit-rates of 0.056 bpp and 0.110 bpp with average PSNR of 33.51 dB and 31.14dB on the "Miss America" and "Salesman" sequences respectively [37]. The most significant feature of their coding algorithm was that the blocking effects were removed. Thus this video coder outperformed all existing spatial domain fractal video coders.

### 5.4.2 Wavelet-Based Fractal Approximation

This video coding system, presented by Kim, C. and Kim, N. [38], utilizes progressive fractal subtree coding and a motion estimation/motion compensation technique.

#### 5.4.2.1 Algorithm

The first step of this compression scheme is to perform some kind of ME/MC on each frame of the video sequence. Thereafter the residual error between the original and MC frame is transmitted to the fractal subtree coding sub-system. A similar algorithm to Algorithm 5-1 is used to code this residual frame. The parameters from both the ME/MC component and the fractal subtree coded portion are stored. The frame memory is refreshed by decoding both the above sub-components.

#### 5.4.2.2 Performance and Efficiency

This coder has revealed results to outperform the H.263 video compression standard, in terms of compression. They have shown an increase of about 0.45 dB in average PSNR over the H.263 standard on the "Miss America" sequence [38].

## 5.5 Summary

This chapter initially presented a wavelet analysis of fractal image coding. The results indicate that up to the DC component, the fractal block coder is equivalent to the fractal-wavelet subtree coder if the Haar wavelet basis is used. The use of a smooth wavelet basis, in the subtree coder removes the tiling artefacts present in fractal block coders. A general fractal-wavelet subtree image compression algorithm was presented. This algorithm required that all wavelet coefficients below some detail level to be stored. Thus the fractal method acts as a prediction of higher frequency coefficients (of the wavelet transform) from lower frequency ones.

A comparison of the different fractal-wavelet coders present in literature suggests that the hybrid schemes that combine both wavelet and fractal coding algorithms achieve greater performance than pure fractal subtree coders. It was also noted that the subtree structure provides a new representation of motion structures in video coding algorithms. An advantage of using this subtree representation is that it can describe large object motion activities.

# CHAPTER 6 - PROPOSED VIDEO CODING SYSTEM

The main aim of this project was to develop a low bit-rate video coder that combines the coding techniques of wavelet and fractal systems. Hence, the proposed compression system was built directly from the observations and results presented in Chapter 5. It is of importance that the proposed coder exploits redundancies present in the video sequence. Chapter 5 has shown that the wavelet domain produces a subtree structure that characterizes motion activities. Furthermore, fractal coding in the wavelet domain removes the tiling effect that is normally associated with fractal coders. Thus all video coding is performed in the wavelet domain.

Section 6.1 will provide a brief overview of the compression and decompression systems of the proposed video codec. This gives a basic idea of the different sub-components involved in the coding process. The next section will analyse each sub-component in detail and account for the reason of choice in each case.

## 6.1 System Overview

This section presents a general idea of the proposed system that is used for low bit-rate video coding. The block diagram of this video system is shown in Figure 6-1. This system takes a similar approach to the variable tree size video compression system (Section 5.4.1 [37]). Each video frame is captured from a video source and transferred to the system. The result of this action provides a raw formatted matrix of the captured video frame. Only the luminance component of each video frame is considered. However, this algorithm can easily be extended to compensate for the chrominance components as well.

The first step is to produce a multi-resolution pyramidal wavelet decomposition of the video frame. The motion activities at different layers of the wavelet pyramid are different but highly correlated since they actually characterize the same motion structure. The frame is split into subtrees, which provides a new method of representing the motion structure. Since consecutive frames are highly correlated, this scheme exploits the redundancy between frames, by forming the domain subtrees in the previous frame. Thus, the range subtrees can be constructed in the current frame, with the inclusion of the coefficients at the lowest frequency subbands. The coefficients at the coarsest scale now form the tree nodes. Also note that the range and domain subtrees are of the same length, which makes it easier for matching comparisons.

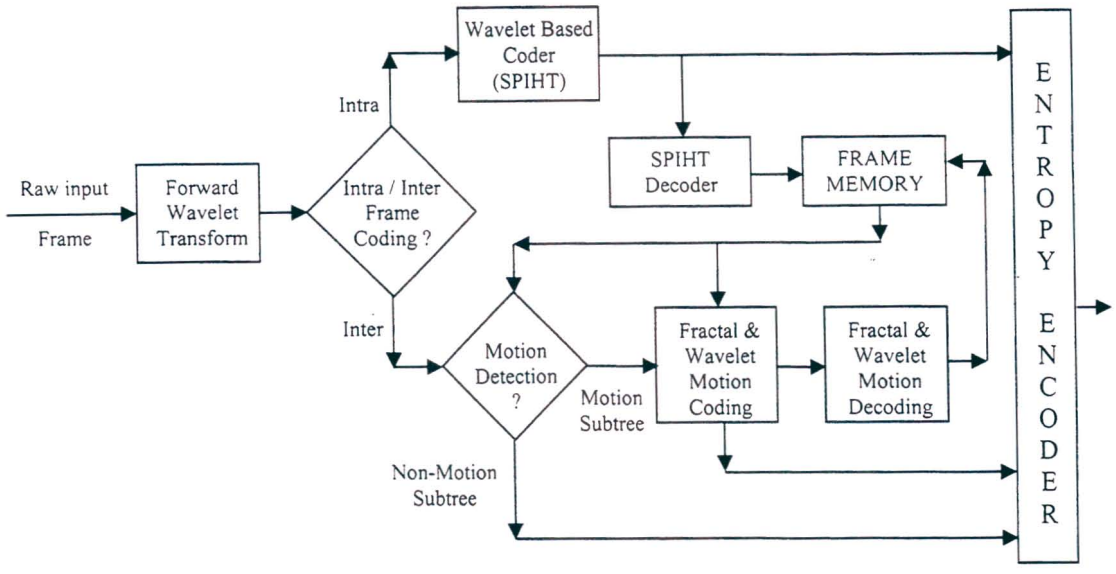



Figure 6-1: Block diagram of proposed video encoder

This algorithm is split into two parts namely intra-frame and inter-frame coding. Intra-frame coding involves the coding of the frame by itself, using image compression techniques. In this scheme the SPIHT algorithm is chosen to perform the intra-frame coding. The first frame of a sequence is always intra-frame coded.

Inter-frame coding is the coding of a frame using data from previous frames. Since a large portion of the previous frame will be repeated in the current frame, it is not required to code the entire frame. If we can separate the motion and non motion portions, a large reduction in the bit-rate will occur. Thus it is a requirement that a motion detection (MD) or motion estimation criterion be used. The ME/MC technique is applied in the wavelet domain. The output stream produced from both the intra and inter frame coding algorithms are input to the entropy encoder.

The decoding process, depicted in Figure 6-2, is simpler and much faster. The bit-stream produced from the encoder is sequentially applied to the entropy decoder. This produces a set of decisions that informs the decoder to produce a representation of the original video sequence.

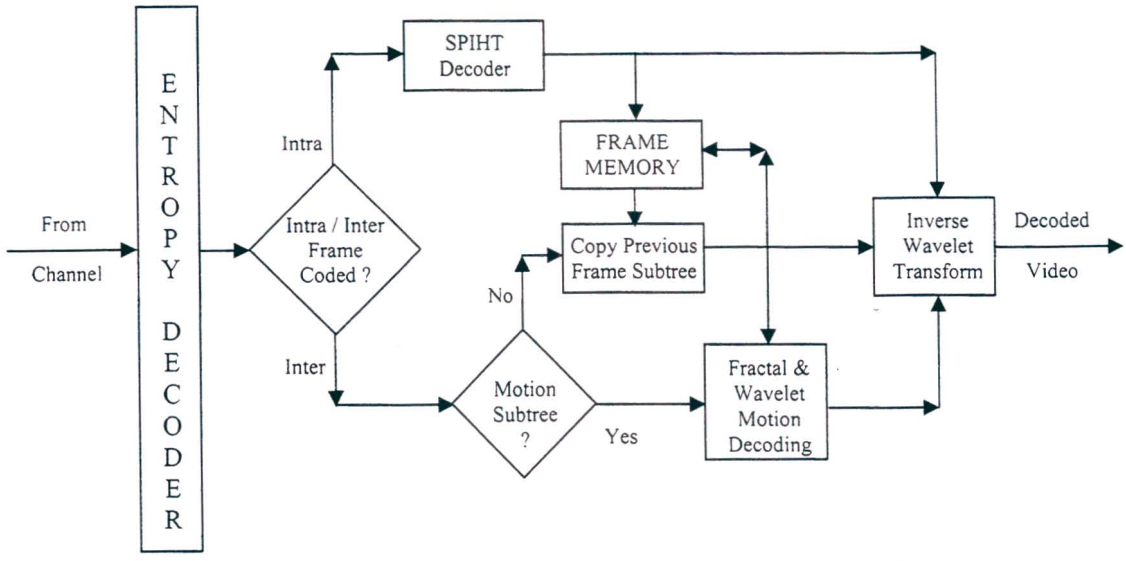



Figure 6-2: Block diagram of proposed video decoder

## 6.2 Details of the Proposed Algorithm

This section is dedicated in examining the details of the different components in the proposed video coding system. Furthermore, justifications are also provided in each case to motivate the use of different components.

### 6.2.1 The Header

As a requirement with all multi-format compression schemes, a header file must initially be transmitted (or stored). The header file consists of a sequence of bits that informs the decoder of the different parameters used at the encoding stage. For this scheme, these parameters include the resolution (or size) of the original sequence, the number of levels of the wavelet transform and the frequency of the intra-frame coding.

The number of bits representing each of these parameters can be dramatically reduced by specifying predefined choices. For example, this coding scheme supports two video sequence formats, namely CIF and QCIF, which must be represented in the YCrCb video format. Thus the resolution of the original image can be stored with only one bit. Similarly the number of levels of the wavelet transform computed is also stored with one bit since the maximum level that can be constructed with the QCIF format is four, and that of CIF is five. Thus the choice of four or five level decomposition is only provided for the CIF format. As explained in Section 1.1.2, it is vital that intra-frame coding be performed at regular intervals to reduce the amount of

error propagating through the system. Once again specific intervals are specified. In this case four intervals are chosen, and these are coded with two bits.

It can therefore be seen that the cost of including the header is insignificant in that it only costs a maximum of four bits. The choices presented in parameter changing are not much. However, this coding system has the possibility of personalization, by providing further user-defined options. Providing these options will increase the size of the header, but will still remain insignificant with respect to frame coding.

### 6.2.2 Transform Stage

The variable tree size fractal video coder, developed by Zhang; (Section 5.4.1) has shown superior performance over all spatial domain fractal video coders. Thus fractal coding in the wavelet domain for video sequences was considered. Furthermore spatial domain fractal video coders have shown to produce the blockiness effect (or tiling structure) in the decompressed sequence. Owing to the discussion presented in Section 5.1, it was shown that fractal coding in the wavelet domain can remove this tiling effect, by using smooth wavelet basis.

Two design choices arise from performing the wavelet transform. These are the type of wavelet basis to use and the number of decomposition levels to perform. Section 4.2.3 has shown that the most likely choice of the wavelet basis is the Daubechies 9/7 wavelet. The other wavelet basis that also provides good performance in compression algorithms is the bi-orthogonal 4.4 (bior4.4) wavelet. Thus both of these wavelet bases are used to perform the wavelet decomposition of each video frame. As described earlier, the maximum allowable decomposition levels of a QCIF video sequence is four and for the CIF is five. Let us assume, without loss of generality, that the video format is QCIF and the number of decomposition levels is four.

For the Daubechies 9/7 wavelet, the lifting method presented by W. Sweldens [77], is used to compute the wavelet transform. On the other hand, the Mallat's filter-bank method, described in Section 4.1.3.2, is used to calculate the wavelet transform using the bior4.4 wavelet.

Since the wavelet transform is performed first, all video coding is performed in the wavelet domain. Zhang et al [37] have shown that the motion activities at different layers of the wavelet pyramid are different but highly correlated since they actually characterize the same motion structure. Furthermore, they have motivated through simulation results of their video coder, the success of the subtree structure in performing ME/MC.

### 6.2.3 Intra-Frame Coding

Intra-frame coding is performed using a still image compression technique. Throughout Chapters 3, 4, and 5 several image compression algorithms were discussed. A comparison of the best coders in each chapter is presented in Figure 6-3. It can be seen that fractal image coders do not compete with current wavelet based compression schemes in terms of compression performance and complexity. Wavelet-based fractal coders have improved performance on the spatial domain fractal coders, but still have increased computational complexity over wavelet ones.

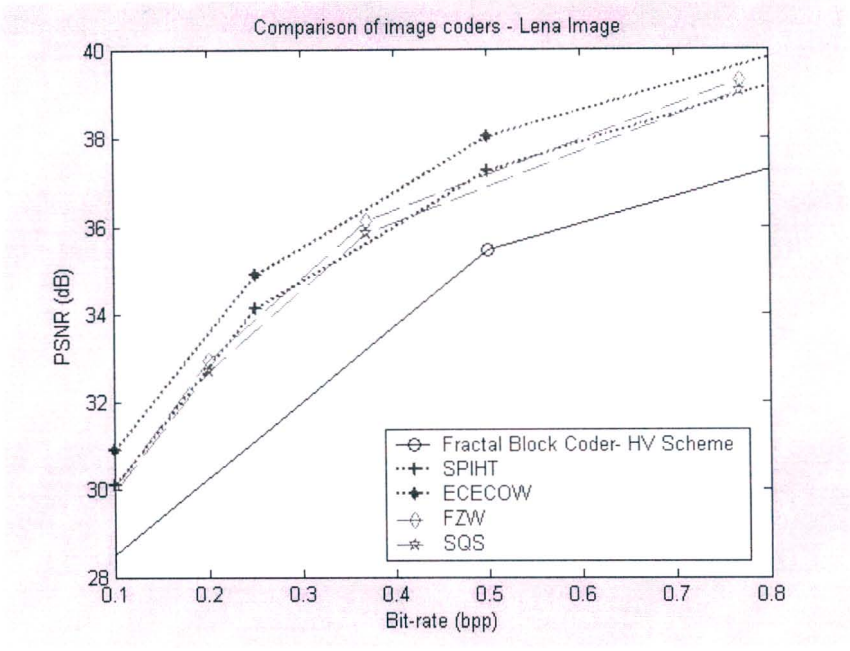



Figure 6-3: Performance comparison of the best Fractal, Wavelet and Fractal-Wavelet coders

From the above comparisons, two wavelet based algorithms that have a good performance to complexity ratio were considered. These are the SPIHT and the ECECOW algorithms. The ECECOW algorithm concentrates all its efforts in the entropy coding of the image, and this differs from the standard adaptive arithmetic coding scheme, which is used in the proposed coding scheme. Thus a redesigning of the ECECOW algorithm needed to be performed in order to combine these two entropy coding systems. Furthermore, the SPIHT zerotree coding structure is similar in representation to the subtree structure. As will be seen in Section 6.2.4, a combination of subtree and zerotree coding can easily be achieved.

For these reasons, the SPIHT algorithm is used in the intra-frame coding block. The first frame in any video sequence is always intra-frame coded, thus providing the highest bit representation. The amount of allowable error present in the intra-frame is considered since this affects the successive coding of inter-frames. Also, intra-frame coding is carried out after a specific

interval. This is done as to refresh the reference frame such that errors do not propagate though the system.

### 6.2.4 Inter-Frame Coding

#### 6.2.4.1 Motion Estimation/Motion compensation

Temporal redundancies can be removed by ME/MC algorithms. In video sequences, there are two main types of motion namely global motion and local motion. Global motion causes the entire video frame to be altered. This is normally caused by camera motion, which includes panning, tilting and zooming. Local motion, on the other hand affects certain areas within a frame, such as an object motion over a constant background.

Global motion is much more difficult to account for. There exist global ME/MC techniques that account for frame co-ordinate changes, with the majority taking the 3-D video coding approach. Since we are concentrating on intra/inter frame coding, global ME/MC will not be considered.

Local ME/MC techniques are either based on block matching or object modelling methods. The object modelling approach extracts a moving object from a scene and predicts its motion throughout the scene. This type of modelling is beyond the scope of this dissertation and will not be explored any further. The block matching technique is normally performed in the spatial domain. Here, frames are divided into blocks, and each block from the current frames is matched with a block from the previous frame. This forms a kind of forward predictive coding. The obvious outcome of this block based ME/MC is that the recovered sequence suffers from the tiling effect.

Block based ME/MC is very similar to spatial domain fractal video coders. Therefore to remove the tiling effect, it is natural to extend spatial domain block based ME/MC to the wavelet domain. The analogy presented in Section 5.1.3, suggests that a block in the spatial domain corresponds to a wavelet subtree, together with a coefficient in the coarsest scale. Thus block based ME/MC is extended to subtree based ME/MC and this will be explained in the following sections.

#### 6.2.4.2 Motion Detection

Motion detection is a form of motion estimation where subtrees that contain motion structures are separated from subtrees with non-moving structures. The motion detection criterion is based on the mean square error (MSE) [37]. Here, a subtree in the current frame is compared to the

same subtree in the previous frame. Note that the coarsest scale coefficients corresponding to that particular subtree are also used in the comparison. If the MSE between the two subtrees is less than a predefined threshold, then the subtree is a non-motion subtree and can be coded with a binary decision i.e. one bit needs to code this subtree. In the decoding process, if the subtree is a non-motion subtree, then the subtree from the previous frame is copied to the new frame. On the contrary, if the MSE is greater than the threshold, then this subtree needs to be coded by another technique. The technique proposed is described next.

#### 6.2.4.3 Coding of Motion Subtrees

The coding of the motion subtrees is divided into two parts. The first part involves fractal coding of the motion subtree. In particular, the variable tree size algorithm is used (Section 5.4.1). The coarsest scale node is removed from the tree and scalar quantized. The domain search pool is formed from the previous frame subtrees minus the coarsest scale root node (at level 4 of the transform). Now this motion coder must find the best matching transformed domain subtree for a given range subtree.

The distortion (given by the MSE), between the best matching transformed domain subtree and range subtree, is then compared to a threshold. If the distortion is less than the threshold, then the subtree is fractal quantized and the transformation parameters as well as the position of the domain subtree are stored. If the distortion is greater than the threshold, the three nodes of the subtree are removed and scalar quantized. The subtree is split into four children subtrees with nodes now at level 3.

The new domain subtrees are constructed from the previous frame, with nodes also at level 3. For each of the children subtree, a best matching transformed domain subtree is sought. If the distortion is less than a predefined threshold, that child subtree is fractal quantized and encoded as its parent. However, if the distortion is greater than the threshold, the co-ordinates of the three nodes of the child subtree are added to the LIP and LIS for SPIHT encoding. If three or more child subtrees do not find a matching domain subtree within a given threshold, the entire subtree from level 4 is SPIHT encoded, with the root node co-ordinate being added to the LIP and LIS.

This second portion of this scheme involves the implementation of the SPIHT algorithm. The standard SPIHT algorithm is performed on the LIP and LIS, after all the subtrees have been fractal coded. No initialisation of the sets is required, since elements have been added to these sets at the fractal coding stage.

#### 6.2.4.4 Performance and Complexity Improvements

To reduce the computational complexity, the nearest neighbour search is performed for the fractal coding method. In other words, surrounding transformed domain subtrees are searched (from the previous frame), to match the corresponding range subtree (subtree to be coded). This should reduce the search time, without substantially affecting the performance since object motion between frames will not change much.

The other enhancement can be made in the SPIHT algorithm. In order to reduce the magnitudes of the wavelet coefficients, the difference between the current frame and previous frame is calculated. Experimental results suggest there is a performance gain in the bit-plane coding of the SPIHT algorithm.

### 6.2.5 Frame Memory

A frame memory is required to store the previous frame, which is used at the inter-frame coding stage. Two approaches can be taken here, in that the frame memory could store the original frame or the decoded frame. Storing the original frame has the advantage of a better ME/MC. However, at the decoding stage, the original frames are not present as the reference frames. Hence, this will significantly affect the decoded image quality, since the reference frame is now the previous decoded frame. Thus the approach taken is to use the decoded frame as the reference. Therefore, after each frame coding is completed, the frame is then decoded and stored in memory as a reference to code the next frame. The frame memory refreshes periodically with the newly decoded frame

### 6.2.6 Entropy Coding

The efficiency of any coding method can be improved by entropy coding its output. For this stage we had a choice between Huffman and Arithmetic coders. Section 2.1.2 details the advantages of arithmetic coding over Huffman coding. Thus the arithmetic coder was chosen, specifically the adaptive arithmetic coder. Said and Pearlman [48] have showed that there is some performance gain achieved in implementing arithmetic coding with the SPIHT algorithm. For further details on the implementation of the adaptive arithmetic the reader is referred to [60].

## 6.3 Summary of the Proposed Coding Algorithm

This chapter has proposed a new video compression system. The system comprises mainly of a combination of fractal and wavelet coding methods, presented in Chapters 3, 4 and 5. In this chapter, each important component of the compression system is singled out, explained in detail and accounted for.

The proposed coder exploits temporal redundancy that is present in video sequences. Hence the algorithm was based on the intra/inter frame video coding approach. This chapter has shown that the best intra-frame coder to use is the wavelet SPIHT algorithm due to the very low computational complexity of this compression method. Furthermore, this algorithm is easily adapted to other components of the proposed video codec.

The majority of the design effort was placed in designing the inter-frame coder, since it is here that most of the bit-rate reduction occurs. This sub-system combined techniques from the variable tree size fractal video compression algorithm and the SPIHT coder. All coding was performed in the wavelet domain. A motion detection component was added that allows the inter-frame algorithm to process only on the motion portions of each frame. This consequently provided a large saving in bit allocation. Motion compensation was achieved by the combined efforts of the variable tree size fractal scheme and the SPIHT method. The SPIHT method adds detail to the recovered image in cases where the fractal approximation fails.

The other elements of the proposed system, such as the wavelet transform and the arithmetic coder, are standard. Details of these algorithms are provided in the referenced literature. One of the major advantages of developing a fractal video coding method in the wavelet domain, is that the tiling (or blocking) artifacts, present in most video compression standards, are removed.

# CHAPTER 7 - PERFORMANCE OF PROPOSED CODER

This chapter is dedicated to assess the complexity and performance of the proposed video coder that combines fractal and wavelet techniques. Several well known test sequences are used to obtain the results. This test set, chosen from the set of standard MPEG video test sequences, consists of a variety of natural scenery and different levels of zoom. The assessment of the computational complexity is a vital requirement in the design of new video coders. It has been well documented that fractal based video coders tend to be computationally expensive due to the block matching methodology. Thus tests of the overall system computational complexity as well as the complexity of different sub-components are made. Furthermore, a comparison of the performance of this new system against the video compression standards is provided.

Section 7.1 will provide a description of the platform and testing method used in assessing the performance of different coders. Section 7.2 will use the platform defined above and carry out different tests on the proposed algorithm, as well as on video compression standards. The testing is executed on different video sequences characterizing various motion activities. It is in this section, that comparisons are made between the proposed algorithm and the H.263+ and MPEG-4 compression standards. The comparisons are defined in terms of R-D performance and computational complexity. The following section provides further analysis on the individual components of the proposed codec. Finally Section 7.4 discusses all the results obtained during the testing process.

## 7.1 Experimental Method

It is of importance that in order to compare the complexity of different video coding algorithms, they must be computed on similar platforms. Hence all results, for the proposed coder as well as for the video compression standards, were obtained on an Intel Pentium-4, 2.4 GHz PC platform. This system was equipped with 256 MB of RAM, running the Windows XP operating system. The simulations of the proposed coder on different test sequence were executed from within the "Microsoft Visual C++" environment.

The H.263+ coder was obtained from the University of British Columbia [75]. The coder configurations, used for all test sequences, appear in Table 7-1. This is the low complexity H.263+ mode. The MPEG-4 coder [76] was utilized with the default settings. This version of

the codec did not allow the user to customize the options, which was noted in their user manual. Thus compression at only one bit-rate could be achieved. Most of the time, this bit-rate was set at 40 kbit/s. It must be noted that the results obtained for both compression standards is slightly inferior to those published in literature. This is due to the coder settings not being optimized for rate-distortion performance, but rather for low computational complexity.

| **Quantization Parameter** | 13 |
|---|---|
| **Motion Vector Mode** | H.263+ |
| **Syntax Based Arithmetic Coding** | Enabled |
| **Advanced Prediction Mode** | Enabled |
| **Deblocking Filter** | Enabled |
| **Advanced Intra-coding Method** | Enabled |
| **Reference Picture Selection Mode** | Enabled |

Table 7-1: H.263+ configuration settings [59]

The computational complexity of the video coding standards were analysed using "Microsoft Profiler", which is a part of "Microsoft Visual Studio". This program was successful in measuring the time spent in individual functions by the specified coder. This consequently gave an indication of which sub-systems in the algorithm required the most amount of processing power.

The video sequences used in the test were chosen from the MPEG standard test sequences. All of these were in the QCIF format i.e. a frame size of 144x176. Processing was only performed on the luminance component (greyscale processing), in which each pixel is represented by 8 bits. The frame rate of these sequences is 30 fps, with duration of 10 seconds.

## 7.2 Comparison with Video Compression Standards

This section compares the performance of the proposed coder against the H.263+ and MPEG-4 coders, using various test sequences. In each sequence, the type of motion structure is varied. For each comparison, the rate-distortion measurement is provided, as well as the encoding times. Furthermore, the $50^{th}$ and $100^{th}$ recovered image frame is presented. Since this is a low bit-rate video coder, the frame rate of the proposed coder is reduced to 10 fps unless otherwise stated. It has been noticed that in the video compression standards, the majority of processing is spent in the ME/MC functions. This section will exhibit the percentage of time these coders spend in the ME/MC algorithms.

### 7.2.1 Akiyo

#### 7.2.1.1 Sequence Characteristics

The Akiyo sequence is a video of a news reader in a studio. The camera, which is fixed, is focussed on the reader. This sequence consists of limited local motion structures, in which the head, facial features and shoulders move. There is no background motion. An example of the frames of the sequence is provided in Figure 7-1.

(a) (b) (c)

Figure 7-1: Video frames of the original "Akiyo" sequence
(a) 50th frame (b) 100th frame (c) 150th frame

#### 7.2.1.2 Performance and Efficiency

The proposed video coder has comparable compression times with respect to the compression standards (see Table 7-2). Specifically at low bit-rates, the complexity of the proposed scheme is less than the video compression standards, but as the bit-rate increases, the processing time increases. This is due to the proposed coder trying to add more resolution detail to the compressed video, after the fractal compression is performed. It is also noted that since this sequence does not contain a lot of motion, all coders spend the majority of the time in the ME/MC algorithms. The R-D performance of the proposed coder is superior to the standard algorithms, achieving consistently a recovered image quality of greater than 1 dB (Table 7-3). Furthermore, this superiority is illustrated graphically in Figure 7-2. Figure 7-3 provides a numerical representation of the recovered image quality achieved on a frame by frame basis.

| | Proposed | H.263+ | | MPEG-4 | |
|---|---|---|---|---|---|
| Bit-rate (kbit/s) | Coding Time (s) | Coding Time (s) | % ME/MC | Coding Time (s) | % ME/MC |
| 15 | 15.83 | 18.49 | 85.33 | X | X |
| 20 | 13.90 | 24.80 | 83.47 | X | X |
| 25 | 17.46 | 24.97 | 82.94 | X | X |
| 40 | 26.18 | 23.72 | 86.08 | 18.656 | 89.3 |
| 50 | 28.54 | 24.61 | 84.69 | X | X |

Table 7-2: Complexity comparison - "Akiyo"

| | Proposed | | H.263+ | | MPEG-4 | |
|---|---|---|---|---|---|---|
| **Bit-rate (kbit/s)** | **Avg. PSNR (dB)** | **fps** | **Avg. PSNR (dB)** | **fps** | **Avg. PSNR (dB)** | **fps** |
| 15 | 32.41 | 10 | 30.49 | 7.1 | X | X |
| 20 | 32.87 | 10 | 31.35 | 9.8 | X | X |
| 25 | 34.51 | 10 | 32.13 | 10.6 | X | X |
| 40 | 37.39 | 10 | 34.40 | 11.3 | 34.44 | 10 |
| 50 | 37.97 | 10 | 35.18 | 11.5 | X | X |

Table 7-3: R-D performance comparison - "Akiyo"


Figure 7-2: Recovered frames at 40 kbit/s - "Akiyo", Frame 50, 100 and 150

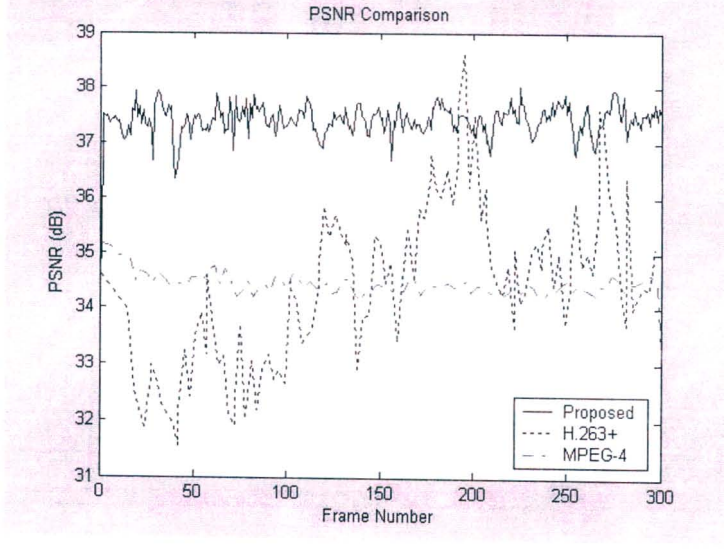
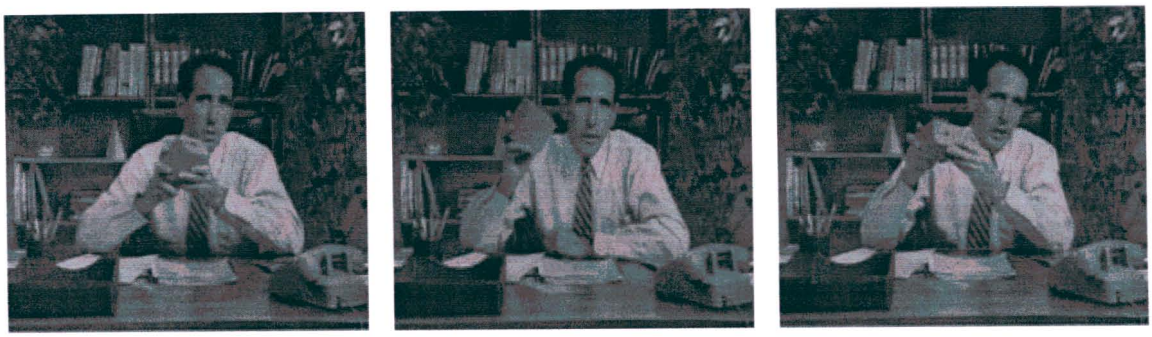



Figure 7-3: Frame by frame PSNR for "Akiyo" sequence at 40 kbit/s

### 7.2.2 Salesman

#### 7.2.2.1 Sequence Characteristics

In this sequence the person in the video attempts to sell the item present in his hand. The camera remain fixed and hence there is no global or background motion. However, this video comprises of local foreground motion, in which the persons head, body and hands shift.



(a) (b) (c)

Figure 7-4: Video frames of the original "Salesman" sequence
(a) 50th frame (b) 100th frame (c) 150th frame

#### 7.2.2.2 Performance and Efficiency

For this sequence, the H.263+ coder seems to have a slight improvement over the proposed coder. However, the computational expense is greater. The larger amount of local motion in this sequence accounts for the poorer image quality as compared for the "Akiyo" sequence. Due to this motion, more bits are allocated in the SPIHT component of the inter-frame coder, since it is more difficult to find a good matching domain subtree. The R-D results are presented in Table 7-5, Figure 7-5 and Figure 7-6.

| | Proposed | H.263+ | | MPEG-4 | |
|---|---|---|---|---|---|
| Bit-rate (kbit/s) | Coding Time (s) | Coding Time (s) | % ME/MC | Coding Time (s) | % ME/MC |
| 20 | 18.36 | 35.85 | 80.23 | X | X |
| 25 | 20.81 | 35.07 | 82.57 | X | X |
| 40 | 25.48 | 33.96 | 84.28 | 24.39 | |
| 50 | 26.47 | 32.46 | 43.26 | X | X |

Table 7-4: Complexity comparison - "Salesman"

| | Proposed | | H.263+ | | MPEG-4 | |
|---|---|---|---|---|---|---|
| Bit-rate (kbit/s) | Avg. PSNR (dB) | fps | Avg. PSNR (dB) | fps | Avg. PSNR (dB) | fps |
| 20 | 30.25 | 10 | 30.66 | 11.0 | X | X |
| 25 | 31.33 | 10 | 31.40 | 11.1 | X | X |
| 40 | 33.54 | 10 | 33.11 | 11.5 | 31.83 | 10 |
| 50 | 34.16 | 10 | 34.21 | 11.6 | X | X |

Table 7-5: R-D performance comparison - "Salesman"


Figure 7-5: Recovered frames at 40 kbit/s - "Salesman", Frame 50, 100 and 150

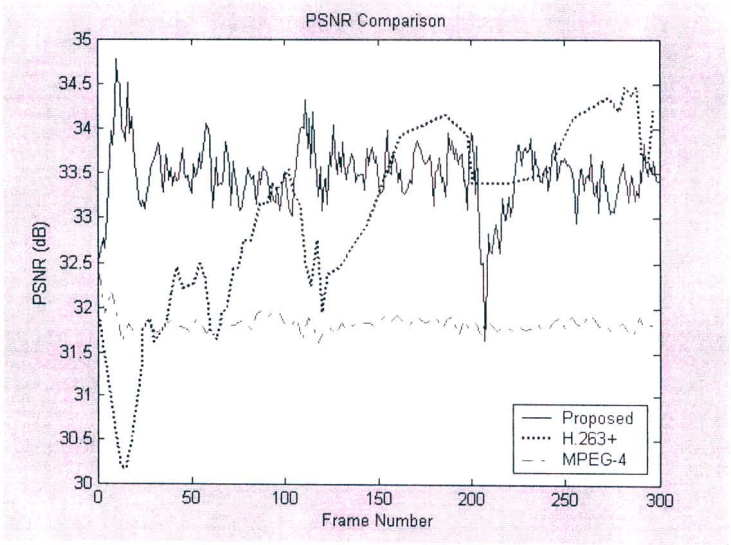



Figure 7-6: Frame by frame PSNR for "Salesman" sequence at 40 kbit/s

### 7.2.3 News

#### 7.2.3.1 Sequence Characteristics

The "News" sequence consists of two news readers, together with a video clip playing in the background. In fact, the "Akiyo" sequence is a different camera angle of this video. Once again, the camera is fixed and there is no zoom. The foreground motion consists of the head, face and shoulder movements, while the background video clip contains a dance sequence by two ballet dancers. The ballet dance is filmed by a camera, which first focuses on both dancers. After a period of time the focus shifts to one of the dancers.


(a) (b) (c)

Figure 7-7: Video frames of the original "News" sequence
(a) 50th frame (b) 100th frame (c) 150th frame

#### 7.2.3.2 Performance and efficiency

From the results for the "Akiyo" sequence, the proposed coder achieves very good performance on parts of the sequence that just contain facial and shoulder movement. Thus the foreground of the "News" sequence is encoded efficiently, which can be seen from the recovered frames in Figure 7-8. Hence, the majority of the bit allocation of the proposed coder accounts for the background motion. Since the motion of the ballerina is not localized, once again it would be difficult to find matching domain subtrees. Hence more bits will be allocated from the SPIHT component of the algorithm, which reduces the compression ratio.

The performance is the proposed coder is still better than the video compression standards, in terms of complexity and R-D performance.

| | Proposed | H.263+ | | MPEG-4 | |
|---|---|---|---|---|---|
| Bit-rate (kbit/s) | Coding Time (s) | Coding Time (s) | % ME/MC | Coding Time (s) | % ME/MC |
| 20 | 20.96 | 30.80 | 87.32 | X | X |
| 25 | 21.84 | 31.09 | 82.77 | X | X |
| 40 | 26.33 | 30.96 | 85.64 | 27.23 | 88.69 |
| 50 | 30.28 | 30.00 | 79.50 | X | X |

Table 7-6: Complexity comparison - "News"

| | Proposed | | H.263+ | | MPEG-4 | |
|---|---|---|---|---|---|---|
| **Bit-rate (kbit/s)** | **Avg. PSNR (dB)** | **fps** | **Avg. PSNR (dB)** | **fps** | **Avg. PSNR (dB)** | **fps** |
| 20 | 29.23 | 10 | 28.93 | 10.7 | X | X |
| 25 | 30.92 | 10 | 29.91 | 11.1 | X | X |
| 40 | 31.65 | 10 | 31.53 | 11.1 | 32.63 | 10 |
| 50 | 33.01 | 10 | 32.45 | 11.6 | X | X |

Table 7-7: R-D performance comparison - "News"

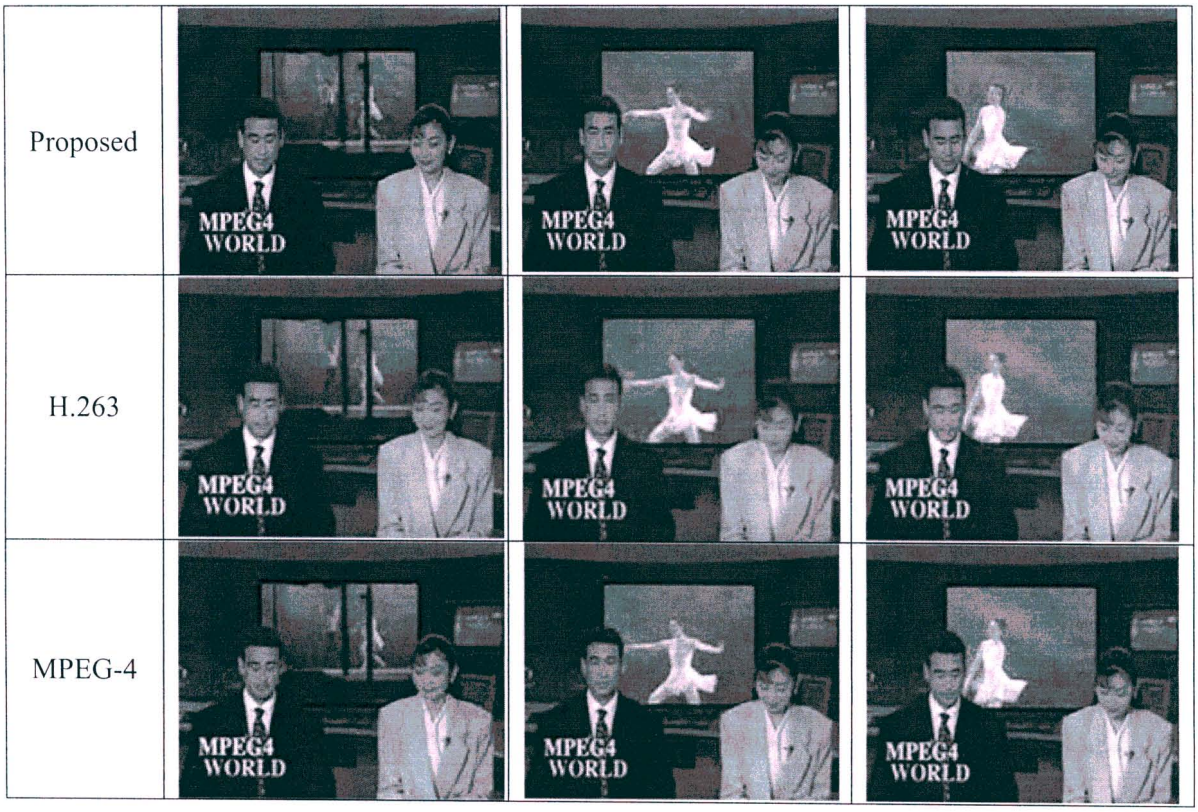



Figure 7-8: Recovered frames at 40 kbit/s - "News", Frame 50, 100 and 150

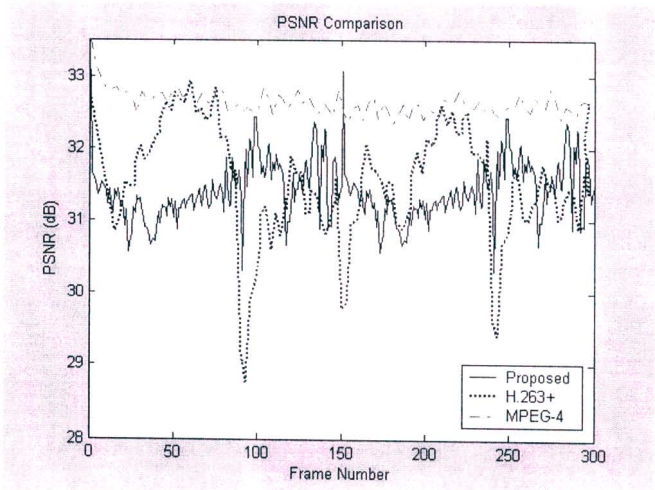
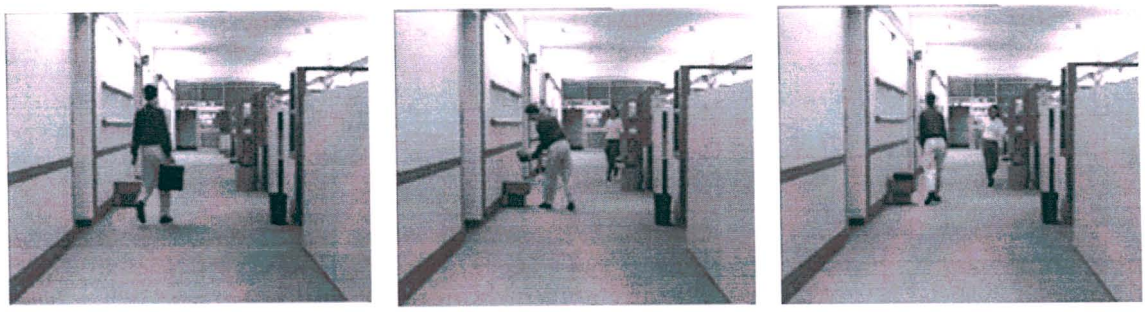



Figure 7-9: Frame by frame PSNR for "News" sequence at 40 kbit/s

### 7.2.4 Hall-Monitor

#### 7.2.4.1 Sequence Characteristics

The "Hall monitor" video sequence is a surveillance camera scene. This sequence consists of huge amount of local motion, in which objects appear and disappear. Furthermore, local motion comprises of full body movement. Here again the camera is fixed, and thus there is no global motion.



(a) (b) (c)

Figure 7-10: Video frames of the original "Hall-Monitor" sequence
(a) 50th frame (b) 100th frame (c) 150th frame

#### 7.2.4.2 Performance and Efficiency

Once again the proposed coder reveals superior performance. In this sequence, the motion of the two people up and down the hall is localized, in that there is not much rapid motion. Furthermore, there is more self-similarity present in sequence, since the motion is in a relatively small area of the frame. It is only in portions of the sequence, in which the human objects appear, that the coder does not act efficiently. This is illustrated in Figure 7-12 by the dips in the PSNR. The R-D performances appear in Table 7-9 and Figure 7-11.

| | Proposed | H.263+ | | MPEG-4 | |
|---|---|---|---|---|---|
| Bit-rate (kbit/s) | Coding Time (s) | Coding Time (s) | % ME/MC | Coding Time (s) | % ME/MC |
| 20 | 19.38 | 24.10 | 82.95 | X | X |
| 25 | 21.25 | 24.49 | 82.82 | X | X |
| 40 | 25.00 | 24.93 | 85.03 | 17.22 | 82.27 |
| 50 | 28.70 | 23.91 | 79.79 | X | X |

Table 7-8: Complexity comparison - "Hall-Monitor"

| | Proposed | | H.263+ | | MPEG-4 | |
|---|---|---|---|---|---|---|
| **Bit-rate (kbit/s)** | **Avg. PSNR (dB)** | **fps** | **Avg. PSNR (dB)** | **fps** | **Avg. PSNR (dB)** | **fps** |
| 20 | 31.31 | 10 | 29.64 | 9.8 | X | X |
| 25 | 31.80 | 10 | 29.92 | 10.1 | X | X |
| 40 | 33.88 | 10 | 31.51 | 11.2 | 33.22 | 10 |
| 50 | 34.70 | 10 | 32.60 | 11.3 | X | X |

Table 7-9: R-D performance comparison - "Hall-Monitor"


Figure 7-11: Recovered frames at 40 kbit/s - "Hall Monitor", Frame 50, 100 and 150

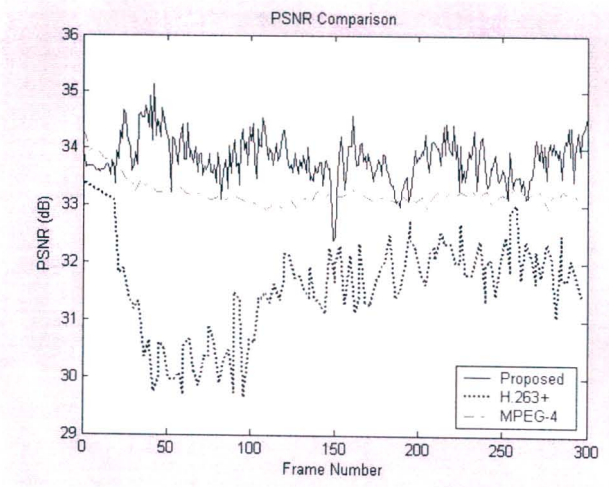
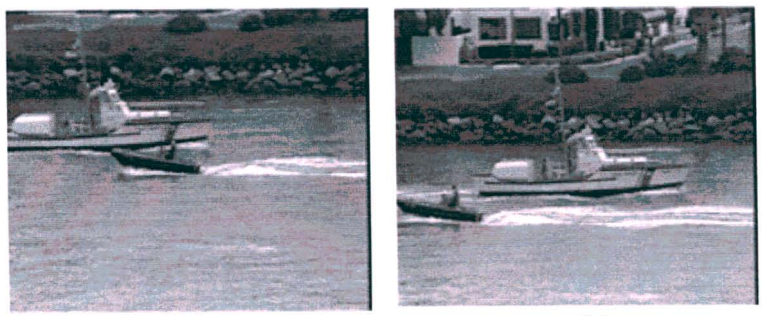



Figure 7-12: Frame by frame PSNR for "Hall Monitor" sequence at 40 kbit/s

### 7.2.5 Coast Guard

#### 7.2.5.1 Sequence Characteristics

The "Coast Guard" video consists of a scene in which there is both camera and object motion. The video is shot from another boat, which moves along with the larger boat. Thus there are large amounts of global and local motion structures.



(a) (b)

Figure 7-13: Video frames of the original "Coast guard" sequence
(a) 50th frame (b) 100th frame

#### 7.2.5.2 Performance and Efficiency

For this particular sequence, the proposed coder operates poorly. This is due to the large amounts of global motion present. For global motion, the idea of the inter-frame coder fails since the background is continually changing. The fractal component of the coder does not succeed in coding many of the range subtrees, due to the lack of self-similarity. Thus the majority of the effort in subtree coding is spent in the SPIHT algorithm, accounting for very low compression ratios. Hence to achieve the high compression ratio, the image quality suffers.

The MPEG-4 standard accounts for global motion by increasing the bit-rate, while maintaining the image frame quality. This is due to the lack of bit-rate control present in the coder that we employed. Hence this method does not hold well for low bit-rate coding. The H.263+ on the other hand, reduces the frame of the sequence to improve the R-D performance. These results are depicted in Table 7-11.

Since the proposed algorithm does not have an automatic frame rate adjustment, the frame rate was manually adjusted to 3 fps so that an equal comparison with the H.263+ coder could be accomplished. This modified algorithm produces a coded sequence with an average PSNR of 26.67 dB, and a coding time of 12.43 s. Thus the proposed coder still outperforms the H.263+ standard.

| Bit-rate (kbit/s) | Proposed Coding Time (s) | H.263+ Coding Time (s) | H.263+ % ME/MC | MPEG-4 Coding Time (s) | MPEG-4 % ME/MC |
|---|---|---|---|---|---|
| 20 | 59.72 | 29.19 | 41.49 | X | X |
| 25 | 36.74 | 38.17 | 46.36 | X | X |
| 40 | 46.79 | 53.51 | 47.34 | X | X |
| 50 | 31.92 | 43.72 | 47.67 | X | X |
| 308 | X | X | X | 70.109 | 81.03 |

Table 7-10: Complexity comparison - "Coast Guard" [59]

| Bit-rate (kbit/s) | Proposed Avg. PSNR (dB) | Proposed fps | H.263+ Avg. PSNR (dB) | H.263+ fps | MPEG-4 Avg. PSNR (dB) | MPEG-4 fps |
|---|---|---|---|---|---|---|
| 20 | 22.23 | 10 | 25.84 | 1.2 | X | X |
| 25 | 23.36 | 10 | 25.68 | 1.2 | X | X |
| 40 | 23.82<br>26.67 | 10<br>3 | 25.34 | 2.6 | X | X |
| 50 | 24.09 | 10 | 25.42 | 3.7 | X | X |
| 263 | X | X | X | X | 30.845 | 10 |

Table 7-11: R-D performance comparison - "Coast Guard" [59]

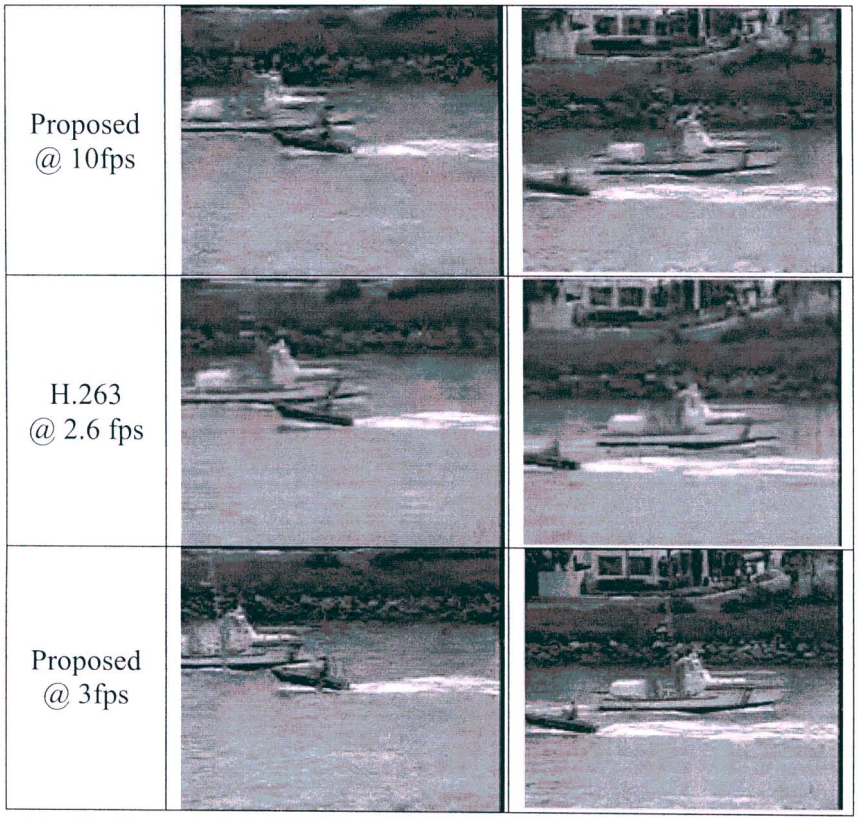



Figure 7-14: Recovered frames at 40 kbit/s - "Coast-Guard", Frame 50 and 100

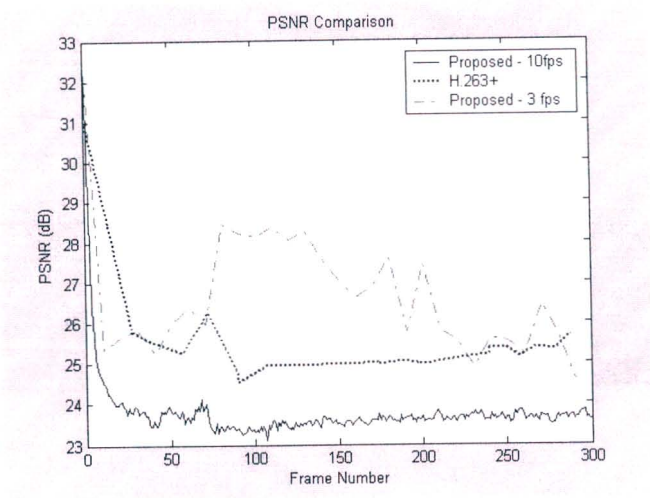



Figure 7-15: Frame by frame PSNR for "Coast Guard" sequence at 40 kbit/s

## 7.3 Further Analysis of Proposed Coder

### 7.3.1 Decoding Times

Table 7-12 provides a list of decoding times computed by the proposed coder on different test sequences. Comparing these times to the coding times presented in Section 7.2, illustrates that the decoding process is much quicker and less complex. This occurs since the computationally expensive subtree searching method is no longer required. Furthermore, the coding stage also decodes each frame as to update the frame memory. So in essence, the coding process provides both the coded and decoded versions of the sequence. It must also be noted that the iterative decoding of fractal methods no longer applies since the domain subtrees are constructed from the previous frame.

The decoding time for almost all the sequences, at different bit-rates, is consistently maintained in a range between 8.5 and 9.5 seconds. This is sufficient to provide real time decoding, since the original sequence length is 10 seconds.

| | 20 kbit/s | 25 kbit/s | 40 kbit/s |
|---|---|---|---|
| **Video Sequence** | **Decoding Time (s)** | **Decoding Time (s)** | **Decoding Time (s)** |
| **Akiyo** | 8.469 | 8.984 | 9.140 |
| **Salesman** | 9.625 | 8.681 | 9.157 |
| **Hall Monitor** | 8.214 | 8.328 | 8.829 |
| **Coast Guard** | 9.141 | 8.990 | 9.218 |

Table 7-12: Decoding times of proposed coder on different test sequences

### 7.3.2 Choice of Wavelet Basis

As was alluded to in Section 6.2.2, two types of wavelets namely the bior4.4 and the db9/7 were chosen to perform the wavelet transform. Table 7-13 provides a comparison of the R-D performance of the proposed coder using the different wavelet transforms. For both the wavelet basis, the coder settings were unchanged. The results indicate, both wavelet basis exhibit similar performances, with the bior4.4 gaining a slight edge over the db9/7 wavelet.

| Video Sequence | Bit-rate (kbit/s) | Filter Type | PSNR (dB) | Encoding Time (s) |
|---|---|---|---|---|
| Akiyo | 25 | bior 4.4 | 34.51 | 17.47 |
| | | db 9/7 | 34.39 | 18.89 |
| | 40 | bior 4.4 | 37.39 | 26.18 |
| | | db 9/7 | 37.03 | 25.86 |
| Salesman | 25 | bior 4.4 | 31.13 | 20.81 |
| | | db 9/7 | 30.69 | 17.94 |
| | 40 | bior 4.4 | 33.54 | 25.48 |
| | | db 9/7 | 33.19 | 25.5 |
| Hall-Monitor | 25 | bior 4.4 | 31.8 | 21.25 |
| | | db 9/7 | 32.09 | 21.75 |
| | 40 | bior 4.4 | 33.88 | 25.00 |
| | | db 9/7 | 33.71 | 23.27 |
| Coast Guard | 25 | bior 4.4 | 23.36 | 36.74 |
| | | db 9/7 | 23.45 | 35.39 |
| | 40 | bior 4.4 | 23.82 | 46.79 |
| | | db 9/7 | 23.93 | 44.36 |

Table 7-13: Comparison of different wavelet basis used in the proposed coder

### 7.3.3 Individual Component Complexity

This section assesses the computational complexity of the individual components in the proposed coding system. The results are presented in Table 7-14, Table 7-15 and Table 7-16. These tables indicate the amount of time each sub-algorithm processes.

The wavelet transform stage on an average takes about 1.2 s to compute for an entire sequence. This consists of the executing of the forward and reverse wavelet transform. In all of the experiments performed, intra-frame coding was performed only on the first frame of the sequence. This consequently accounts for the low computational time. It must be noted that this time indicates the processing speed of the SPIHT algorithm on a frame.

In all three tables, the majority of the processing power is spent in the motion detection, motion estimation and compensation algorithm. This is to be expected, as this is the core of the coding system.

The arithmetic coding time are small, and increases slightly with an increased bit-rate. This occurs since more symbols are output by the coding algorithm.

| | 25 kbit/s | | 40 kbit/s | |
|---|---|---|---|---|
| | Time (s) | % | Time (ms) | % |
| **Total** | 13.90 | 100 | 26.18 | 100 |
| **Wavelet Transform** | 1.187 | 8.53 | 1.235 | 4.72 |
| **Inter-Frame** | 10.90 | 78.41 | 23.90 | 91.29 |
| **Intra-Frame** | 0.093 | 0.66 | 0.093 | 0.36 |
| **Arithmetic Coder** | 0.061 | 0.44 | 0.094 | 0.35 |
| **Other** | 1.659 | 11.94 | 0.858 | 3.27 |

Table 7-14: Sub-component timing - "Akiyo"

| | 25 kbit/s | | 40 kbit/s | |
|---|---|---|---|---|
| | Time (s) | % | Time (ms) | % |
| **Total** | 20.81 | 100 | 25.48 | 100 |
| **Wavelet Transform** | 1.172 | 5.63 | 1.218 | 4.78 |
| **Inter-Frame** | 17.47 | 83.95 | 23.36 | 91.68 |
| **Intra-Frame** | 0.062 | 0.30 | 0.094 | 0.37 |
| **Arithmetic Coder** | 0.079 | 0.38 | 0.201 | 0.79 |
| **Other** | 2.027 | 9.74 | 0.607 | 2.38 |

Table 7-15: Sub-component timing - "Salesman"

| | 25 kbit/s | | 40 kbit/s | |
|---|---|---|---|---|
| | Time (s) | % | Time (ms) | % |
| **Total** | 21.25 | 100 | 25 | 100 |
| **Wavelet Transform** | 1.125 | 5.29 | 1.156 | 4.62 |
| **Inter-Frame** | 19.735 | 92.87 | 22.528 | 90.11 |
| **Intra-Frame** | 0.062 | 0.29 | 0.063 | 0.25 |
| **Arithmetic Coder** | 0.048 | 0.23 | 0.08 | 0.32 |
| **Other** | 0.28 | 1.32 | 1.173 | 4.69 |

Table 7-16: Sub-component timing - "Hall Monitor"

### 7.3.4 Comparison with Variable Tree Size Video Coder

Since the proposed video compression system is based on the variable tree size video coder (Section 5.4.1), a performance comparison is made in this section. The published results of the variable tree size coding system reveals obtained bit-rates of 0.056 bpp and 0.110 bpp with averaged PSNR of 33.51dB and 31.14dB on the "Miss America" and "Salesman" test sequences, respectively. On the same two sequences with the same achieved bit-rate, the

proposed coder acquires PSNR of 34.32 dB and 31.53 dB, in that order. Thus, the proposed system achieves a higher compression performance than the variable tree size coding system. The performance gain can be accounted for by noticing that in areas of the video sequence where the fractal approximation is poor, the wavelet component of the proposed system adds more detail. This consequently increases the recovered frame quality, with no significant increase in the bit-rate.

## 7.4 Discussion of Results Obtained and Conclusion

The proposed video coder has shown excellent results when being compared to existing video compression standards. In the majority of the test sequences, the coder outperforms these standards. One of the major advantages of performing fractal coding in wavelet domain, is that the tiling effect is removed. This effect is more prominent in the H.263 and MPEG-4 standards, especially at low bit-rates. Thus the recovered images computed by the proposed algorithm are much clearer.

An interesting observation from the complexity tables in Section 7.2 is that as the bit-rate increases, the compression time increases for the proposed system. This takes place as a result of more detail being added to each frame, by the SPIHT algorithm at the inter-frame coding stage. For low bit-rates, the minimum threshold set in the SPIHT algorithm is high. This consequently results in less least significant bits of the wavelet transform coefficients being stored. Hence the recovered image quality is low, but the compression is high. Higher bit-rates cater for more least significant bits to be stored. This subsequently entails a greater coding time.

The proposed system achieves maximum benefit on sequences that contain few local motion structures, such as the "Akiyo" sequence, and in video where motion occurs in a small area of the frame, as in the "Hall-monitor" sequence. The main reason for this is that a majority of the previous frame, such as backgrounds, is repeated in the current frame. Thus these features are efficiently coded by the motion detection criteria. Furthermore, whatever local motion is left over contains some sought of self-similarity and is therefore coded well by the fractal portion of the scheme. Any other residue is processed using few bits by the SPIHT algorithm. As the local motion become complex ("Salesman" and "News"), more coding is completed by the SPIHT method. This results in an increased bit-rate and processing time. It is observed in the frame by frame PSNR figures, that for sequences with few local motion structures, there PSNR per frame does not fluctuate. This happens since there are less errors propagating through the system, and hence maintaining the required bit-rate.

In video that contain global motion, the system performance reduces drastically. Such a case is for the "Coast Guard" sequence. Here, the majority of the scheme is spent in the SPIHT processing, rather than the fractal systems. Although self-similarity regions may exists in these sequences, the nearest neighbour searching method employed in the fractal subtree matching restricts these regions to be in close proximity of the subtree being coded. Increasing the search area significantly increases the coding time. A solution to the global motion problem was to use a manual frame rate control, in which the required frame rate is reduced. Results on the "Coast Guard" sequence showed an enhancement in performance of the proposed system over the H.263+ standard operating at the same frame rate.

Due to the lack of results presented in published papers that contain wavelet based fractal video coders, it was difficult to compare the performance of the proposed coder. However, with respect to variable tree size fractal video coder (Section 5.4.1), the proposed coder does have an improved performance. Furthermore, this coder outperforms all spatial domain fractal video coders by a large margin.

To summarize, the proposed video compression system outperforms existing video compression standards in the majority of the tests conducted. One reason for this is that the low complexity versions of the H.263+ and MPEG-4 codecs were used. The proposed codec operates optimally on sequences that contain local motion structures, producing highly acceptable recovered image qualities at extremely low bit-rates.

# CHAPTER 8 - REAL TIME IMPLEMENTATION OF PROPOSED CODER

In order to provide efficient multimedia services, it is of vital importance that the codec used is capable of achieving the task in real time. This chapter deals with the implementation of the proposed coding algorithm on a DSP chip. Due to limited storage requirements and processing power of the DSP platform, several algorithm optimization techniques were implemented. This consequently reduces the processing time, thereby accomplishing a real time scenario.

Section 8.1 provides a general description of the hardware and software platform of the DSP. Thereafter, Section 8.2 discusses the performance and implementation issues of different sub-systems of the proposed codec. Section 8.3 assesses the performance of the real time system on different sequences characterizing different motion structures. Finally, Section 8.4 provides some multimedia applications that the coder can be used for.

## 8.1 DSP Platform

The proposed algorithm was developed using the TMS320C6000 imaging developer's kit (IDK). The IDK has been manufactured by Texas Instruments to provide a platform for the development of image/video processing applications. The hardware setup of the IDK is depicted in Figure 8-1.

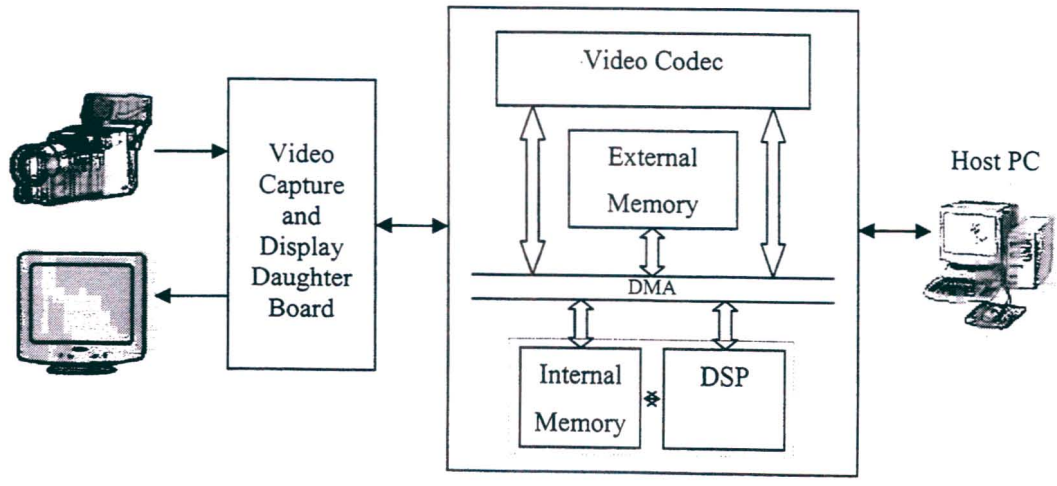



Figure 8-1: Hardware platform of the IDK

The hardware platform consists of four main components: the capture and display daughter board, DSP, host PC communication and the video codec. Each of these components will be detailed below.

a) DSP – The IDK contains the 150MHz, floating point, TMS320C6711 DSP that is capable of executing 900 million floating-point operations per second. Furthermore, the DSP comprises of 64kB internal cache memory as well as 16MB of external synchronous dynamic random access memory. A parallel port controller is included to interface the DSP to a host PC, via the standard parallel port.

b) Capture and display daughter board – The daughter card that is mounted on the TMS320C6711 DSP development board supports both PAL and NTSC systems. A camera is attached to the daughter board through the composite input. Video data capturing and formatting is accomplished by an on-board FPGA that converts the captured frame to the 4:2:2 format in order to separate Y, Cr, Cb components. These components may be sent individually to the DSP for processing. The display driver supports both 8 bpp greyscale and 16 bpp colour (RBG) formats. However, the display provides a maximum resolution of 640x480 or 800x600 on a RGB computer monitor.

c) Host PC communication – The communication module handles communication to and from the host PC via the parallel port. The host port interface provides access to all DSP memory.

d) Video codec – This is the main module of the video coding system. It contains the actual video coding algorithm which is downloaded into the program memory. The software development was carried out using the high powered C compiler and program debugger present in Code Composer Studio ™ version 2. Once the proposed coding algorithm was complied, it is transferred to the DSP using the communication module.

Since real time video compression is always time critical and computationally intensive, this high performance DSP is capable of achieving the compression task. One of the important features of the IDK is the double buffering method that is used to further speed up the processing. In this method, captured data is first brought from external memory to internal memory. The processing is then performed in internal memory. Thereafter the output is stored back in the external memory.

## 8.2 Implementation of the Proposed Coder

This section will highlight the implementation issues and details for each component of the proposed coder.

### 8.2.1 Frame Capturing

A PAL video camera was connected to the IDK. For each captured frame, only the luminance component was considered from the YCrCb format. The frame size was specified to be similar to the QCIF settings. However, the size of PAL frame is 576x768, thus each frame was reduced to 144x192 by local averaging and sub-sampling of neighbouring pixels.

### 8.2.2 Wavelet Transform

The simulation results presented in Section 7.3.2 illustrated that the use of the bior4.4 basis in the wavelet transform achieves better compression results than the db9/7 basis. It was also motioned in Section 6.2.2 that Mallat's fast wavelet transform method was used to compute the wavelet coefficients. Thus, the wavelet transform was implemented using the bior4.4 wavelet basis using the filter bank method.

| No. | Low Pass (Decomposition) | High Pass (Decomposition) | Low Pass (Reconstruction) | High Pass (Reconstruction) |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0.0378 | -0.0645 | -0.0645 | -0.0378 |
| 3 | -0.0238 | 0.0407 | -0.0407 | -0.0238 |
| 4 | -0.1106 | 0.4181 | 0.4181 | 0.1106 |
| 5 | 0.3774 | -0.7885 | 0.7885 | 0.3774 |
| 6 | 0.8527 | 0.4181 | 0.4181 | -0.8527 |
| 7 | 0.3774 | 0.0407 | -0.0407 | 0.3774 |
| 8 | -0.1106 | -0.0645 | -0.0645 | 0.1106 |
| 9 | -0.0238 | 0 | 0 | -0.0238 |
| 10 | 0.0378 | 0 | 0 | -0.0378 |

Table 8-1: Decomposition and reconstruction filter coefficients for the bior4.4 wavelet

The filter coefficients used in computing the bior4.4 wavelet transform is given in Table 8-1. It is immediately noted that these values are floating point, and hence floating point computations are required. Although the chosen DSP supports floating point arithmetic, computational time

can be reduced by executing integer arithmetic. Thus all filter coefficients are scaled up by 15 bits, which is equivalent to multiplying each filter coefficient by $2^{15}$. Hence the filter coefficients are now represented by integers. Once the wavelet transform of an image coefficient is calculated, that transform coefficient value must then be scaled down by 15 bits to bring it to its correct value. Another reason for applying integer arithmetic is that this algorithm can also be implemented on non-floating point DSPs.

The powerful processing capability of the DSP can only be achieved if operations are performed on data that is present in the internal memory. Generally the processing of data in external memory is about 20 times slower that that in internal memory [49]. Thus to improve the overall speed of the algorithm, all core processing should be performed in internal memory. This is normally achieved by the double buffering method.

The size of the internal memory is 64kB, which is sufficient to hold one frame of data (of size 144x192). However, it is definitely not large enough to hold any other computational output of the frame data such as the wavelet transform. A solution to this is to move a block at a time from external memory to internal, perform the wavelet transform of that block in internal memory, and then move the transformed data back to external memory. The architecture of the IDK is designed such that the data moving and data processing can be executed simultaneously. The movement of data is achieved via the direct memory access (DMA) channel.

To take advantage of the double buffering method, two buffers must be allocated for each of the input and output. Thus the internal memory must be split into four equal parts, as shown in Figure 8-2. *Buffer_in* stores the data that has been read from external memory and *Buffer_out* the wavelet transform of the respective *Buffer_in* data. Each data coefficient is stored using 16 bits (short integer format). Thus each buffer in internal memory can store a maximum of 36 rows of data, i.e. the size of a data block is 36x192.

When the computation of the horizontal wavelet transform begins, *Block A* is fetched from external memory, via the DMA channel, and stored in *Buffer_in_0*. When moving the data in *Block B* to *Buffer_in_1*, the wavelet filtering process is being applied to *Buffer_in_0* with the output being stored in *Buffer_out_0*. Both the lowpass and highpass decomposition filters are applied to the data. When the filtering process is completed (for that block), the contents of *Buffer_out_0* are stored in another data array in external memory, i.e. at *Block E*, via another DMA channel. At the same time, the contents of *Block C* are moved to *Buffer_in_0* and the wavelet filtering process begins on *Buffer_in_1*. This process continues until the wavelet decomposition of all data blocks is completed. It must be noted that the contents of the output

buffers are stored column-wise in external memory. This is done such that the same algorithm can be employed to compute the vertical wavelet transform, with the external memory data arrays swapped. After both the horizontal and vertical wavelet transforms are computed, a one level wavelet decomposition of an image frame is performed. The above method can easily be extended to achieve more levels of the wavelet transform.

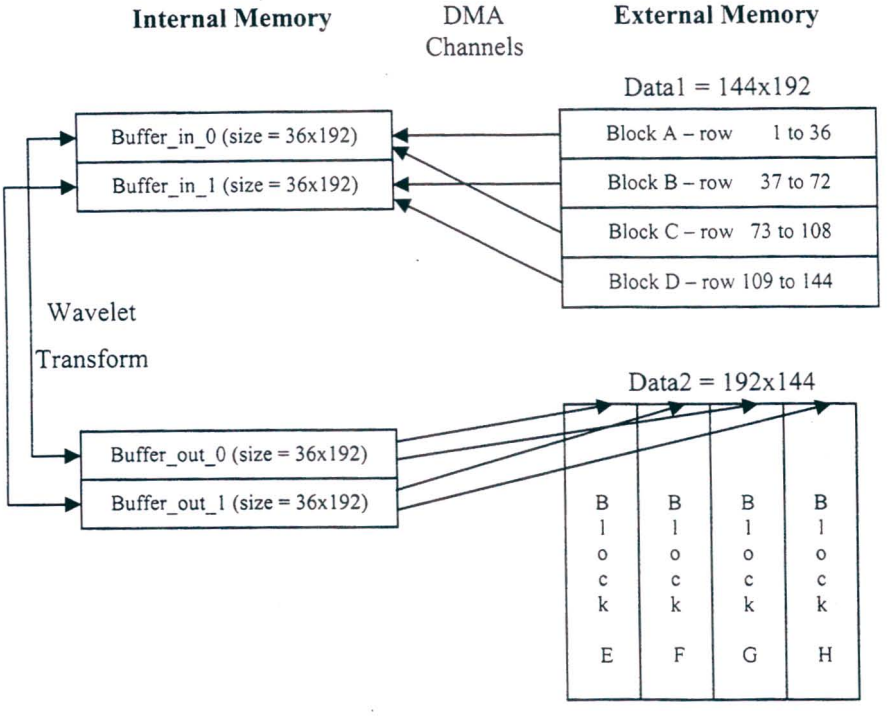



Figure 8-2: Double buffering method to compute the horizontal wavelet transform

### 8.2.3 Intra-Frame Coding Algorithm

The proposed coding scheme utilizes the SPIHT compression algorithm to perform the intra-frame coding. The SPIHT system requires the use of three temporary lists (LSP, LIP, and LIS) to keep track of the significant coefficients and hence improving the codec's efficiency. These lists store the coordinates of different pixels in the wavelet transform. For a frame size of 144x192, each coordinate is stored using 8 bits, giving a total of 16 bits for each entry in any of the lists. The maximum number entries for each list will be the size of the image. This consequently produces a storage requirement of approximately 166 kB. For the current IDK, this is not a serious problem. However, on DSPs with limited external memories or on larger image sizes, this becomes a difficulty. Furthermore, this coding algorithm specifies that the entries in the list must be inserted and deleted. This could cause a large increase in the coding time. For these reasons, a low memory and fast implementation of the SPIHT algorithm was sought.

Sun et al [49] presented a successful low-memory real time implementation of the SPIHT coder. Their coding system removed the LSP and LIP, thereby reducing the memory requirement. Moreover, the refinement pass of the original SPIHT algorithm is no longer necessary and thus reducing the computational complexity. Thus, a similar optimization approach of the SPIHT coding scheme was undertaken.

Since exact bit-rate control of intra-frame coding is not essential, the LSP from the SPIHT algorithm can be removed. To achieve this, a number of error bits ( $\mu_e$ ) is specified, such that if a coefficient is found to be significant, all bits of that coefficient, except for the last few (least significant) bits is stored. These least significant bits correspond to the number of error bits. In other words if the threshold quantisation level in the SPIHT algorithm is $T_h$, the number of bits output for a significant coefficient is $(\log_2(T_h)+1-\mu_e)$. Similarly, the LIP can also be removed, so that when a coefficient is to be added to the LIP, its first $(\log_2(T_h)+1-\mu_e)$ bits is output. The coding process terminates when the threshold drops below $2^{\mu_e}$.

Another efficiency improvement made in the SPIHT implementation is to remove all trees from the LIS whose descendents have magnitudes below $2^{\mu_e}$ [49]. These trees are termed absolute zerotrees and would never reach a stage where it will become significant. The removal of these absolute zerotree will reduce the length of the LIS, thereby reducing the coding time.

It is shown in [49] that there is not much difference in terms of R-D performance between the low complexity version and the original SPIHT coders. However, there is a significant memory reduction by 13-14 times. Moreover, the coding time is greatly reduced.

### 8.2.4 Motion detection

To carry out the motion detection operation, the wavelet transform coefficients of each frame are rearranged into two sets. The first set contains all coefficients from the coarsest scale of the transform. The next set includes all subtrees formed from roots at coarsest scale. Each subtree is constructed by placing the corresponding horizontal, vertical and diagonal quadtrees in that particular order. This new subtree representation of the wavelet transform makes the motion detection process much easier, and also facilitates efficient coding of motion subtrees.

Each subtree in the set is then compared to the equivalent subtree of the previous frame with respect to the MSE. If the computed MSE is greater than a predefined threshold, then the

subtree together with its coarsest scale coefficient are coded as motion subtree. It must be noted that a single bit is used to determine whether a subtree is a motion one or not.

### 8.2.5 Coding of Motion Subtrees

The fractal coding search pool is created from performing the isometries detailed in Table 5-1 on each subtree of the current frame. To reduce the computational time, four isometries were chosen. These include the identity (constructed at the motion detection phase), horizontal flip, vertical flip and 180° rotation operations. The maximum memory expense of the domain search pool is four times the size of storing a frame. However, the proposed algorithm requires that nearest neighbour fractal searching method be employed. Thus the size of the domain search pool can be greatly condensed by limiting the isometry calculation on those subtrees in the vicinity of the motion subtree. Hence the nearest neighbour searching process reduces both computational and complexity expenses.

The rest of the motion coding scheme continues as detailed in Section 6.2.4.3. If the fractal coding of a subtree fails, the three quadtree root nodes are removed and scalar quantized. Thereafter, each of the three co-ordinates is added to the LIS only, for SPIHT encoding. The SPIHT encoding algorithm follows the identical process outlined in Section 8.2.3.

### 8.2.6 Frame Memory

The implementation process described above requires four storage locations that are capable of storing a video frame of size 144x192x16 bits. Two of these memory spaces are used to store the wavelet transform coefficients of the current and previous frame and the other two, to hold the respective current and previous subtree representation. If a new video frame is to be input, the previous wavelet transform memory space is overwritten with the transform coefficients of the new frame. Thereafter the current transform memory buffer becomes the previous reference frame. A similar approach is undertaken for the subtree storage locations.

## 8.3 Results and Performance

This section assesses the performance of the real-time implementation of the proposed video coding system on the IDK. To carry out this assessment, several live video sequences were compressed using this system. These sequences were either captured from a video camera or a television feed. The entire compression system layout is depicted in Figure 8-3.

Figure 8-3: Layout of the video coding system on the IDK

Each video frame captured from the source is transferred to the DSP for compression processing. The size of each frame is 144x192. The respective video frame is then decompressed and displayed on a monitor. After a period of five frames, the average acquired bit-rate is transferred to the host PC. The host PC subsequently produces a graph of the respective bit-rates of the entire video sequence. Also, the compression frame-rate was set to 10 fps.

### 8.3.1 Video Camera Sequences

These sequences were filmed in a seminar room. Each sequence contains a different type of local motion structure. The original and compressed versions of these sequences are included in the attached CD.

#### 8.3.1.1 Sequence One

In this video sequence, a person walks into the camera view, says a few words, and leaves the camera area. The resultant bit-rates obtained for this sequence is shown in Figure 8-5. The majority of this sequence, which contains head, shoulder and facial movements, is compressed efficiently to bit-rates below 40 kbit/s (revealed at time 6 to 13 seconds in Figure 8-5). For the area in which the person walks in and out of the video frame, a higher bit-rate is allocated (shown at time 2 to 4 seconds and 13 to 16 seconds in Figure 8-5). The higher bit-rate accounts for the larger body motion of the person.

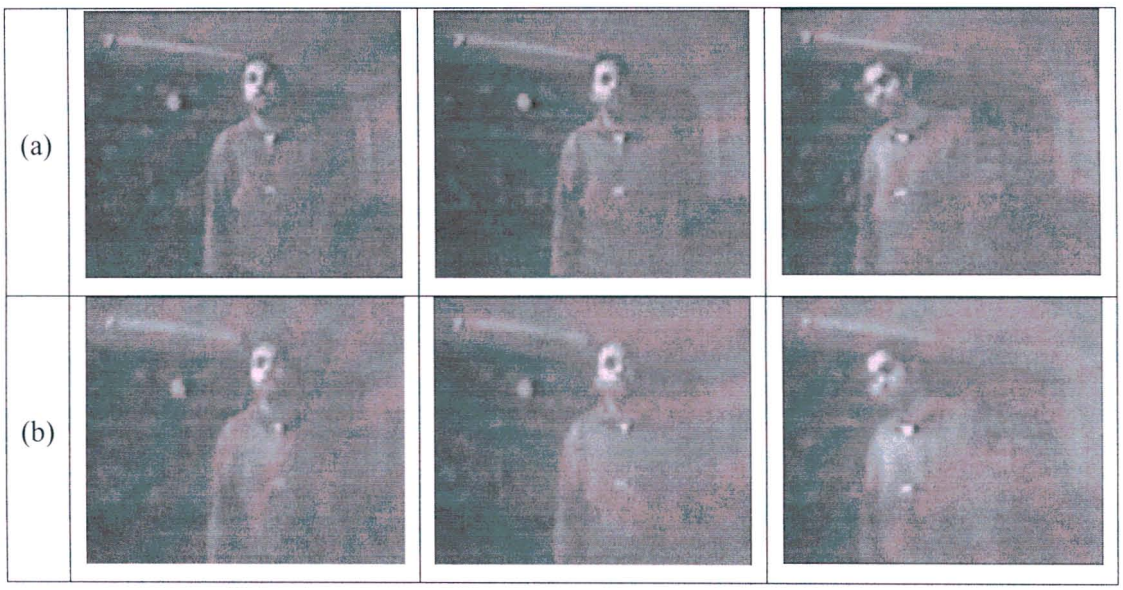



Figure 8-4: (a) Original and (b) recovered video frames for "Sequence One"

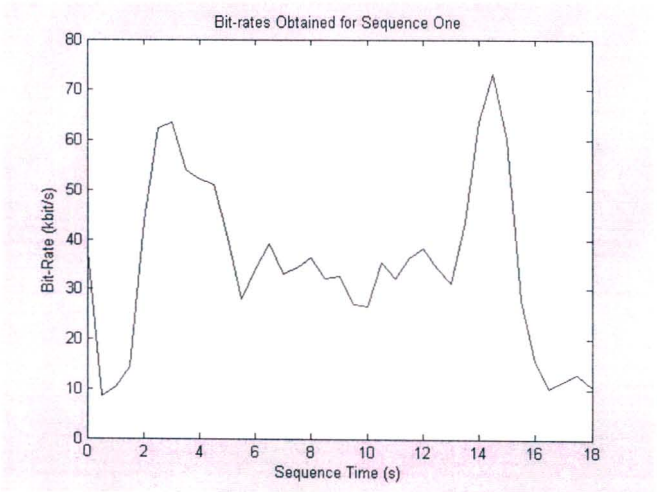



Figure 8-5: Acquired bit-rates for "Sequence One"

### 8.3.1.2 Sequence Two

The second video sequence contains local foreground hand, head and upper body motion. The camera is fixed and is focused on the person. Figure 8-7 illustrates that in areas of larger hand and body movements, the bit-rate achieved is above 60 kbit/s. However, for the most part of this sequence, the bit-rate is maintained below 60 kbit/s.

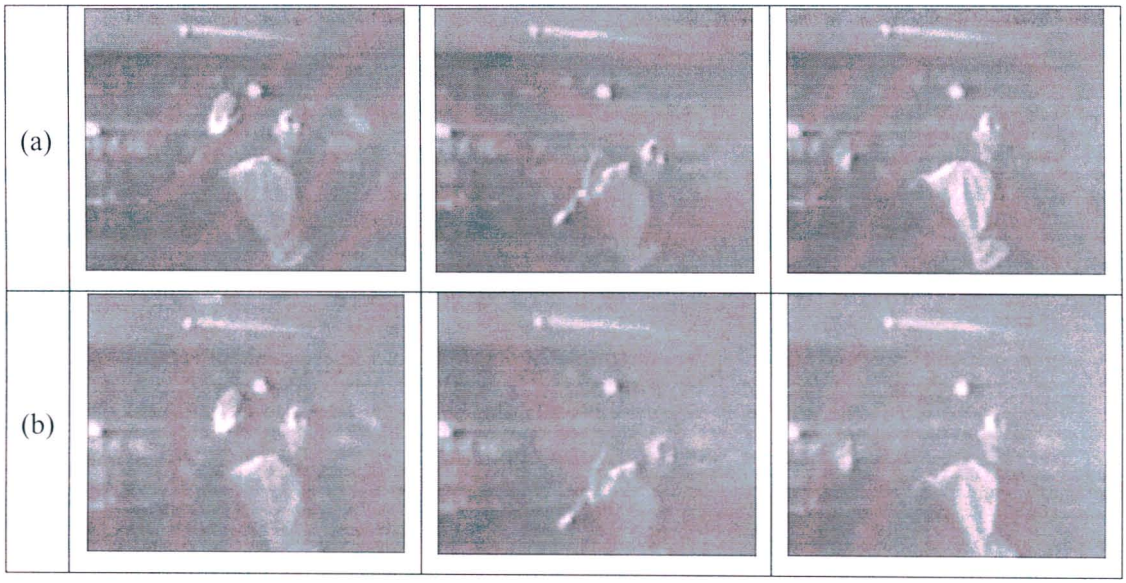



Figure 8-6: (a) Original and (b) recovered video frames for "Sequence Two"

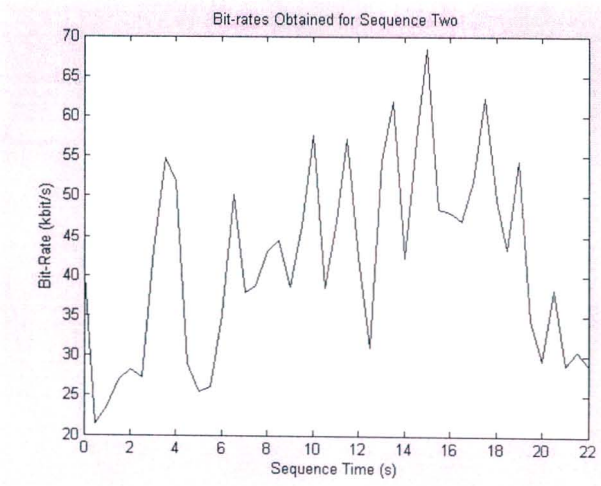



Figure 8-7: Acquired bit-rates for "Sequence Two"

#### 8.3.1.3 Sequence Three

This video comprises of rapid full body motion throughout the sequence. As illustrated in Figure 8-9, the bit-rates attained for this sequence is kept below 65 kbit/s. This sequence illustrates the effectiveness of the proposed coding system in compressing areas of large local motion to low bit-rates, while producing reasonably good reconstructed video frames.

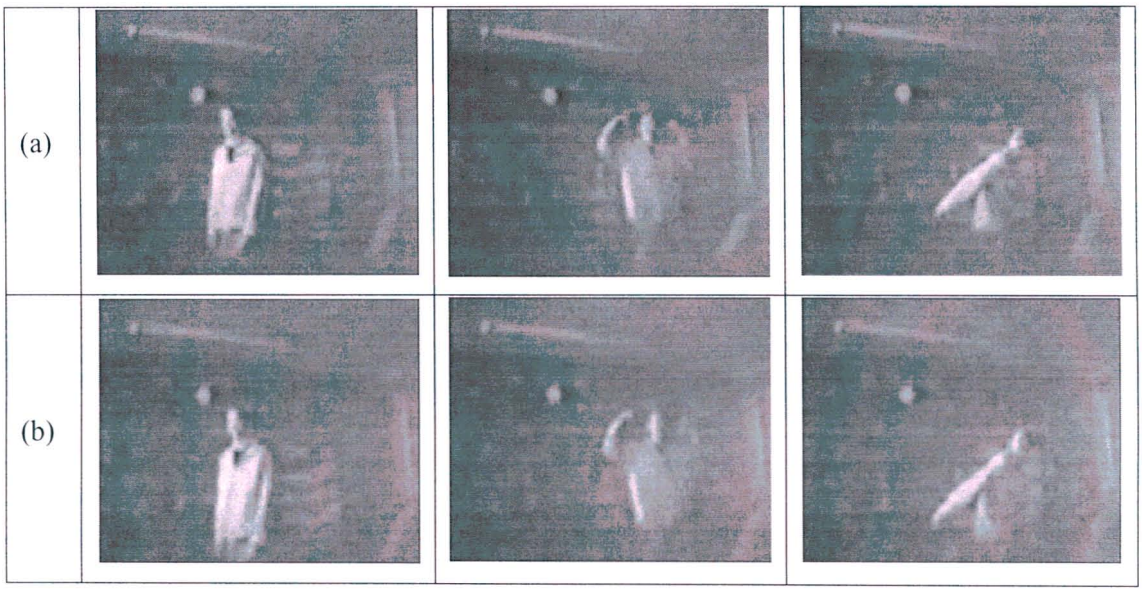



Figure 8-8: (a) Original and (b) recovered video frames for "Sequence Three"

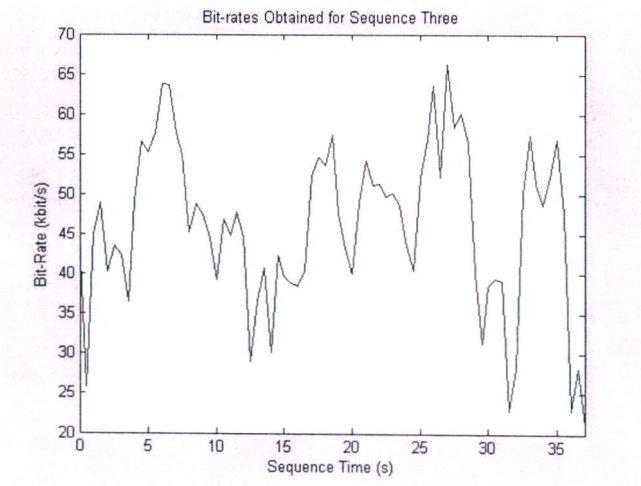



Figure 8-9: Acquired bit-rates for "Sequence Three"

### 8.3.2 Television Sequences

The video sequences obtained for this test were from live television programmes. Since there is some noise present in these sequences (caused by a slightly poor reception), the achieved bit-rates are somewhat higher than what was to be expected.

#### 8.3.2.1 Interview

This video was captured from the "*3rd degree*" television programme on channel "*ETV*". This sequence consists of an interview process, where the camera shifts from the interviewer to the interviewee and vice-versa. The bit-rate map (Figure 8-11) for this particular sequence indicates the successfulness of this compression in compressing a television signal to low bit-rates. The recovered image qualities are also reasonably clear, even in regions where the camera shifts occurs (Figure 8-10).

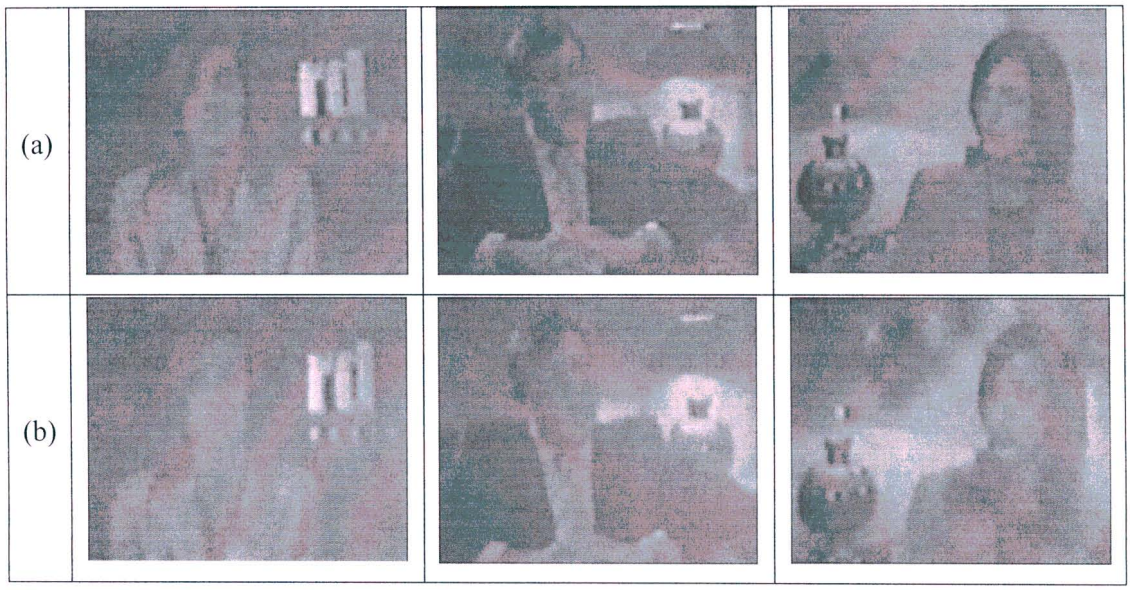



Figure 8-10: (a) Original and (b) recovered video frames for the "Interview" sequence

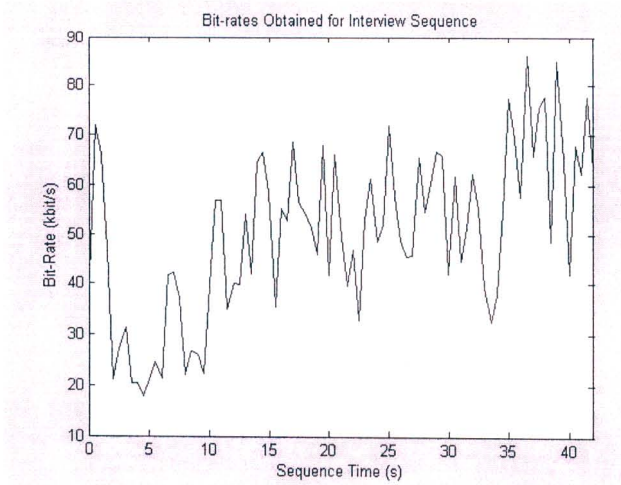



Figure 8-11: Acquired bit-rates for the "Interview" sequence

### 8.3.2.2 Parliament

In this video sequence, an actual recording of a parliamentary speech takes place. This live recording appeared on the "*SABC 3*" television channel. This sequence involves a person giving a speech on a podium and hence containing local foreground motion. Furthermore, there exists local background motion produced by two people situated at a table. Once again, low bit-rates are obtained (Figure 8-13) with very acceptable recovered video images (Figure 8-12).


Figure 8-12: (a) Original and (b) recovered video frames for the "Parliament" sequence

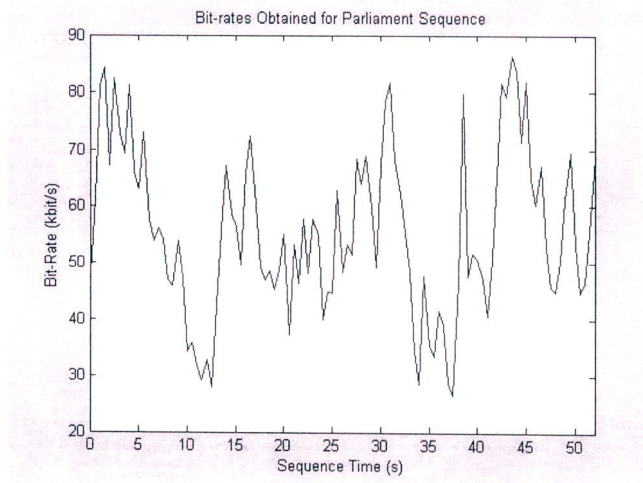



Figure 8-13: Acquired bit-rates for the "Parliament" sequence

### 8.3.3 Individual Component Timings

This section details the computational time achieved by some components of the proposed coding system. Since the frame-rate was set to 10 fps, the entire compression algorithm is performed within 100 ms on each video frame. Table 8-2 includes the average computational time achieved by the different sub-components.

| Sub-component | Computational Time (ms) |
|---|---|
| Forward Wavelet Transform | 19.66 |
| Reverse Wavelet Transform | 22.77 |
| Intra-Frame Coding | 48.86 |
| Inter-Frame Coding | 45.05 |

Table 8-2: Computational time produced by different sub-components.

## 8.4 Multimedia Applications

Technological advancements have prompted the global market to call for new telecommunications facilities that characterize network multimedia applications and projects. These applications mainly consider the communication of images, video and audio. This dissertation has only described compression techniques that focus on images and video. However, there are several efficient audio compression algorithms that do exist in practice. In order to combine both visual and auditory media, it is necessary that the respective compression methods be multiplexed on the same channel. It must be noted that the bandwidth requirement for sound is much less than that of the visual media.

The work presented in this dissertation focussed on low bit-rate compression systems. The reason for this is that the majority of communication structures present in South Africa (and Africa as a whole) is narrowband. Broadband systems are becoming increasingly popular in the market place, but the costs of these systems are still much too expensive. Hence the majority of the population still utilize narrowband systems such as the telephone channel. The maximum achievable bit-rate on such a channel is 64 kbit/s. Thus the visual media should be compressed to a bit-rate of 50 kbit/s or lower to allow for audio to be multiplexed with it.

In general, visual applications can be broken into two categories, namely video communication systems and still picture systems. It has been noted in Section 4.3.6, that the most effective and least complex image compression schemes are the ECECOW and SPIHT algorithms. However, the implementation portion of this project only considered the SPIHT method. Thus to provide still picture applications, the proposed coding system will only operate at the intra-coding stage. Still picture services only require a transmission of an image after a certain amount of time, which is usually in magnitudes of seconds. This consequently implies that more bits can be allocated to the image, while maintaining the same average bit-rate (in kbit/s). Hence the size of the picture can be larger.

Video communication systems, on the other hand, require the frame to be updated several times per second. Normally, for low bit-rate applications, the frame rate is chosen to be 10fps. The proposed coder is capable of obtaining such a frame rate on video sequences that contain local motion structures only. Thus video services will be limited to ones in which local motion is present.

A list of multimedia services or applications will now be detailed.

a) Video Conferencing – Video conferencing systems are aimed to facilitate group based communications. This application allows the transmission of video and audio to groups centred at different venues. The communication can be unilateral or bi-lateral. The type of motion present in such an application will include head, shoulder, hands and facial movement. Hence the proposed system will provide results similar to the 'Akiyo', 'Salesman' and 'News' sequences, depending on the type of local foreground and background motion.

b) Video Streaming – This type of service arises when there is a need to transmit a video feed over the internet. The communication method is normally one-way. The system can be configured to be a multicasting type, in which several users can obtain the video, or dedicated, where one person receives the feed only.

c) Remote Surveillance Systems – Video surveillance systems are used to monitor different areas of a room, passage ways, etc. There is always a need for this information to be transferred to remote locations. In these types of scenes, there is a wide angle of view of people entering and leaving the monitored area. Thus the performance of the proposed system will be similar to that on the 'Hall Monitor' video sequence.

d) Distance Learning – Distance learning is the acquisition of knowledge and skill through communication systems in which the student and the teacher are based in separate locations. This allows the student to attend a real time lecture with the ability of interrupting the lecturer at any time for questioning. This application requires a bi-directional transfer of video and sound. It is also possible to provide other types of teaching material such as presentation slides which requires the transmission of still pictures together with a pointer overlay and audio. This application is much needed in rural areas, where narrowband communication channels are present.

## 8.5 Summary

In this chapter, a real-time implementation of the proposed low bit-rate video compression system on a DSP was realized. This system was able to produce reasonable recovered image qualities on different types of sequences entailing different local motion structures, while maintaining the low bit-rates. In all sequences, a frame rate of 10 fps was managed.

The results acquired in this chapter indicated that this compression system achieves very low bit-rates on sequences containing few local motion structures, such as head and shoulder movements. As the local motion becomes complex, the bit-rate increases, but still remains below 70 kbit/s. Furthermore, this system is effective in compressing sequences containing full body motion. Due to the performance achieved by the proposed video coder, several multimedia applications were listed, where the use of this compression system would be effective.

# CHAPTER 9 - CONCLUSION

This dissertation has presented and discussed several areas of image and video compression. The coding systems investigated were based on well known methods of fractal and wavelets. In this chapter, the important findings and conclusions of each of the preceding chapters is initially presented. Thereafter, future work and recommendations are detailed, which could improve the overall performance of the proposed video coder.

## 9.1 Chapter Summaries

### 9.1.1 Chapter 2 – Standard Types of Compression

In this chapter, the most commonly used media compression system namely the transform based coder was detailed. It was highlighted that the performance of these coders depends on the efficient quantization of the transformed image coefficients. Thus the mathematical transformation utilized in these systems must provide the transformed images with statistical properties that are useful for compression. Further compression can be achieved by employing a lossless entropy coder on the coded information.

Also included in this chapter is an overview of several leading image and video compression standards. All of these standards make use of the transform coding system, with the mathematical transform chosen to be the DCT, except for the JPEG-2000 standard where the DWT is used. The video compression standards, such as the MPEG-4 and H.263, employ block-based ME/MC techniques to compress video sequences. This consequently results in the recovered video producing the blocking or tiling artefacts. The performance analysis of the video compression standards are deferred to Chapter 7.

### 9.1.2 Chapter 3 – Fractals and Iterated Function Systems

In this chapter, the theory of fractal image and video compression methods had been investigated. Fractal image compression resulted from the theory of IFS. The contraction mapping principle guarantees the convergence of an IFS to a unique attractor. So if an image is represented by an IFS, then that image can be reconstructed by iterating the IFS to its attractor. The collage theorem states that in order to find an IFS whose attractor looks like a given image, a set of contractive transformations must be found such that the "distance" between the given

image and the transformations on that image is small. Thus fractal image compression involves finding an IFS for an image.

Fractal coding systems exploit local self-similarity present in the image at the same scale or at different scales. In natural images, this self-similarity property is clear since a part of the image is similar to another part of that same image. The first proficient and automated fractal compression algorithm developed was the fractal block coder, introduced by Jacquin. All subsequent fractal coding schemes are based on the workings of this scheme. This compression system involves the division of an image into blocks of a fixed size, and for each block, a properly transformed copy of another block, in the same image, must be found that matches up to the block in question. The fractal block coding method is efficient in compressing areas of the image that contain straight lines, constant regions and constant gradients, but fails in regions containing textures and unique blocks.

The encoding stage of fractal image coders are typically computational expensive. However, the decoding phase is much faster. Several enhancements have been made to the fractal block coder, by considering different partitioning methods and block searching strategies. In general, fractal coders do not perform better than the state-of-the art image coders in terms of compression ratio and PSNR. All fractal coders produce a characteristic blocking or tiling effect in the decompressed image, especially at high compression ratios.

### 9.1.3 Chapter 4 – Wavelets and the Wavelet Transform

This chapter provides a literature review of the theory of wavelet transforms and wavelet based image compression systems. The wavelet transform provides good energy compaction and decorrelating properties that makes it suitable for compression. Furthermore, the multi-resolution and multi-frequency representation makes it easier for the less significant higher frequency information to be discarded. Wavelet based compression schemes are based on the transform coding system. This chapter finds that by changing the wavelet basis at the transform stage has little or no effect on the overall compression of the system.

Wavelet image coders have illustrated results better than the JPEG image compression standards. The performance gain of wavelet coders occurs in the modelling and representation of the wavelet coefficients. Generally, higher order models, such as the ECECOW algorithm, achieve superior performance. The ECECOW and SPIHT schemes are two of most low complex, high compression wavelet image coders.

### 9.1.4 Chapter 5 – Fractal Compression in the Wavelet Domain

Chapter 5 investigated the ability of combining fractal and wavelet techniques for image and video compression. A wavelet analysis of the fractal block coder illustrated that fractal compression algorithms actually exploit redundancies between different image resolutions, in that it predicts high frequency wavelet coefficients from lower frequency ones. Using this inter-scale prediction method, a general fractal coding scheme in the wavelet domain or fractal-wavelet subtree coder is presented. It is shown that the use of Haar wavelet in the subtree coder is equivalent to the fractal block coding algorithm, up to a DC component.

The major advantage of the subtree coder is that the blocking or tiling artefacts characteristic of fractal block coders can be removed, by using smooth wavelet basis in the computation of the wavelet transform. This consequently improves the recovered image quality at similar compression ratios. Hence the R-D results produced by the fractal subtree coders are greater than the normal fractal compression schemes.

The few hybrid image compression schemes that exist in literature, which combines both the fractal-wavelet subtree and wavelet methods were examined. These hybrid systems achieve performance comparable to state-of-the-art wavelet image coders, but with increased computation complexity. In video compression, the subtree representation can be used to characterize motion structures. The advantage of using this representation is that it can describe large object motion activities instead of piecewise motion activities. The variable subtree size fractal video coder has illustrated the successful implementation of the subtree representation in video compression.

### 9.1.5 Chapter 6 – Proposed Video Coding System

The main contribution of this chapter was to propose a new low bit-rate video compression system that combines fractal and wavelet coding methods. Included in this chapter is a detailed description of each component of the proposed coding scheme.

Having observed the ability of the fractal subtree coder in removing the blocking effect and the effectiveness of the subtree representation in characterising motion activities, all video coding was executed in the wavelet domain. The proposed coding scheme utilized the intra/inter frame video compression system in order to exploit temporal redundancies. All intra-frame coding was performed by the SPIHT image compression method. SPIHT was chosen due to its very low computational complexity and adaptiveness to the inter-frame coding portion of the proposed algorithm.

The inter-frame coding component initially separated motion and non-motion subtrees based on a simple MSE motion detection operation. The resultant motion subtrees were then coded by a unique combination of the variable subtree size fractal scheme and the SPIHT algorithm. The idea of this combination is to add detail to the video frame in cases where fractal compression approximation fails. The final subsystem of the proposed coder consists of the adaptive arithmetic entropy coder which further compresses the coded video data.

### 9.1.6 Chapter 7 – Performance of Proposed Coder

In this chapter, a thorough evaluation of the proposed video coding system is provided. The R-D performance of this new compression system is compared against the H.263+ and MPEG-4 video compression standards using different test sequences. In these experiments, the low complexity versions of the standard coders were used, such that the encoding times matched up to that of the proposed system. The results indicated that the proposed scheme achieves superior results in the majority of these sequences. It is also observed that in compression of these sequences the blocking effect is removed for the new coding system, whereas it is more outstanding in those sequences compressed by the video standards.

The proposed compression system achieves variable R-D results on different sequences that contain different types of motion, which is to be expected. On video that includes few local motion structures, this coder achieves the best compression outcome. This occurs since the majority of the sequence which is stationary, such as the background, is not coded. The motion detection operation is thus successful in eliminating these non-motion portions from being coded. Furthermore, whatever motion sections are left do contain some kind of self-similarity and hence coded efficiently by the fractal component of the coding system. As soon as sequences start containing more local motion, additional coding is performed by the inter-frame section. It becomes increasingly more difficult to fractal code the motion portions, thus increasing the amount of subtrees being SPIHT coded. This subsequently results in an increased bit-rate.

In sequences that contain global motion, such as camera motion, zooming, etc, the performance of the system diminishes. This is due to the coding algorithm not containing efficient global ME/MC techniques. In these types of video sequences, the majority of time is spent at the SPIHT coding stage of the inter-frame subsystem. Regions of self-similarity do exist when global motion is encountered but these regions are not fractal coded. This occurs since most of the time these regions fall outside the fractal search area, given that the nearest neighbour searching strategy is employed, to reduce computational time. To compress global motion

video, the frame-rate of the video was manually reduced. The results indicated that this reduction in frame-rate improved the recovered frame quality and still showed superior performance over the H.236+ standard, operating at a similar frame-rate.

### 9.1.7 Chapter 8 – Real Time Implementation of Proposed Coder

Due to the low computational complexity of the proposed video coding system, a real time implementation on a digital signal processor was possible. This chapter details the complexity reduction techniques performed in order to achieve such an implementation. Specifically, a fast and efficient execution of the filter bank wavelet transform method and the SPIHT algorithm was presented. Although this implementation was performed on the Texas Instruments imaging developer's kit DSP platform, the methods discussed can be used on other platforms, since both coding times and memory cost were considered. The results produced by this implementation on real time video streams indicate that this system performs well on scenes with local motion. Thus, the proposed compression system can be used to provide low bit-rate multimedia services on video applications that contain local motion.

## 9.2 Final Remarks

In this dissertation, a new low bit-rate video compression system, that combines techniques of fractals and wavelets, was proposed. This scheme takes advantage of the wavelet subtree representation of motion structures, in which subtrees containing large spatial domain motion is coded with a fractal variable subtree size coder, and that of small regions of motion, with the wavelet SPIHT encoder. The results obtained by the proposed coder make it suitable for low-bit rate video coding, with the performance of it being comparable, and in some cases superior, to the H.263+ and MPEG-4 video compression standards.

This compression system operates efficiently on sequences with small amounts of local motion, such as in video sequences containing movements of a persons head, shoulder and facial features. In such a sequence, bit-rates of less than 25 kbit/s can be obtained, with very acceptable frame qualities. As more local motion is added to the video, the acquired bit-rates are increased to 50 kbit/s or less. These video sequences contain full body movements. In video containing global motion structures, this system performs poorly. To achieve high compression on these sequences with reasonable frame quality, the frame rate was manually reduced.

In conclusion, the proposed video coding system can find applications using video sequences containing local motion structures. Thus, this system can offer multimedia services, such as

video conferencing, distance learning, live streaming and remote surveillance, on low bandwidth channels.

## 9.3 Future Work

The video compression system proposed in this dissertation has tremendous potential for future work, especially in the area of fractal compression. Several possible coder improvements will be detailed below.

All results presented in this dissertation were achieved using greyscale video and images. Thus, the compression techniques used must be extended to colour sequences by considering the chrominance components. The easiest manner to account for colour is to apply the same compression method (used for the luminance component) to the chrominance components. However, it was noticed that there is significantly more redundancy present in the colour components. Therefore, this redundancy needs to be considered.

At the intra-frame coding stage, the SPIHT algorithm was employed as the encoder. However, Section 4.3.6 has revealed that the ECECOW method produces greater R-D performance than SPIHT, and thus the implementation of the ECECOW algorithm could improve the overall performance of the system. The ECECOW algorithm should also be considered at the inter-frame coding stage as well.

The proposed system utilizes a simple motion detection operation to separate motion and non-motion portions of a frame. Higher order local ME/MC can provide better temporal decorrelation and efficiently separate and encode the moving structures of a sequence. However, these techniques are more complex and therefore needs to be investigated. Furthermore, global ME/MC system is also a requirement to be considered, since the proposed system operates poorly on sequences containing global motion. Global ME/MC techniques are much more computationally expensive and could defeat the whole idea of a low bit-rate system. It is therefore required that a low complexity local and global ME/MC system to be designed.

It is also a requirement of a compression system to be adaptive. This means that if an average bit-rate is required, the frame quality in areas of the sequence producing lower bit-rates need to be increased and that of higher bit-rates be decreased, in order to maintain the average bit-rate. In addition, an automatic frame rate control needs to be introduced; such that the frame rate can be reduced in areas of the video producing very high bit-rates.

Finally, to provide multimedia applications, audio needs to be multiplexed with the compressed video data. Thus an efficient audio compression system needs to be investigated. In real time operation of this multimedia system, possible errors incurred during transmission and reception should be analysed. Furthermore, research into channel error correction schemes should be looked in to.

# References


[1] M.F. Barnsley, "Fractals Everywhere," Academic Press, San Diego, CA, USA, 1988.

[2] J. Kominek, "Advances in Fractal Compression for Multimedia Applications," University of Waterloo, Ontario, Canada, 1997.

[3] M.F. Barnsley and L.P. Hurd, "Fractal Image Compression," AK Peters Ltd., 1993.

[4] A.E. Jacquin, "Fractal Image Coding: A Review," Proceedings of the IEEE, vol. 81, no. 10, pp. 1451-1465, Oct. 1993.

[5] A. E. Jacquin, "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations," IEEE Trans. on Image Proc., vol. 1, no. 1, pp. 18, Jan. 1992.

[6] E.W. Jacobs, Y. Fisher, and R.D. Boss, "Image Compression: A Study of the Iterated Transform Method," Signal Process, Vol. 29, No. 3, pp. 251-263, Dec. 1992.

[7] L.A. Soberano, "The Mathematical Foundation of Image Compression," University of North Carolina, Wilmington, North Carolina, May 2000.

[8] S.K. Alexander, "Two and Three Dimensional Coding Schemes for Wavelet and Fractal-Wavelet Image Compression," University of Waterloo, Ontario, Canada, 2001.

[9] B. Wohlberg, G. de Jager, "A Review of the Fractal Image Coding Literature," IEEE Trans. Image Proc., vol. 8, no. 12, pp. 1716-1729, Dec. 1999.

[10] A. Schuler, "An Introduction to Fractal Image Coding," March 2000.

[11] B. Forte and E.R. Vrscay, "Theory of Generalised Fractal Transforms," University of Waterloo, Ontario, Canada, July 1995.

[12] M.G. Alkhansari, "Fractal Based Image and Video Coding using Matching Pursuit," University of Illinois, Urbana, Illinois, 1997.

[13] M. Pi, "Fractal Image Compression," Department of Computer Sciences, University of Alberta, Alberta, Canada, 2003.

[14] E.K.R. Rao and P.C. Yip, "The Transform and Data Compression Handbook," Boca Raton, CRC Press LLC, 2001.

[15] D. Saupe, "Fractal Image Compression via Nearest Neighbour Search," NATO ASI on Fractal Image Encoding and Analysis, Trondheim, Norway, July 1995.

[16] L. Hurd, M.F. Barnsley and M. Gustavus, "Fractal Video Compression," Proc. of the 37th IEEE Comput. Soc. Int. Conf., vol. 37, pp. 41-42, 1992.

[17] J.M. Beaumont, "Image Data Compression using Fractal Techniques," BT Techn. J., vol. 9, no. 4, pp. 93-109, 1991.

[18] B. Hürtgen and P.Büttgen, "Fractal Approach to Low Rate Video Coding," Proc. of the SPIE: VCIP'93, Boston, Massachusetts, vol. 2094, pp. 120-131, Nov. 1993.

[19] G. Lu and T.L. Yew, "Applications of Partitioned Iterated Function Systems in Image and Video Compression," J. Visual Comm. and Image Rep., pp. 144–154, 1996.

[20] E. Reusens, "Sequence Coding Based on the Fractal Theory of Iterated Transformations Systems," Proc. of the SPIE: VCIP'93, vol. 2094, pp. 132-140, Nov 1993.

[21] M.S. Lazar and L.T. Bruton, "Fractal Block Coding of Digital Video," IEEE Trans. on CASVT, vol. 4, no. 3, pp. 297-308, June 1994.

[22] K.U. Barthel and T. Voyé, "Three-Dimensional Fractal Video Coding," Proc. of ICIP'95, Washington, vol. 3, pp. 260-263, Oct. 1995.

[23] M. Barakat and J.L. Dugelay, "Image Sequence Coding using 3-D IFS," Proc. of ICIP'96, pp. 141-144, Sept. 1996.

[24] T.C. Ferguson and H.R. Wu, "Fractal Transform Techniques for Very Low Bit-Rate Video Coding," IEEE International symposium on circuits and systems, Hong Kong, pp. 1456-1459, June 1997.

[25] A. Pentland and B. Horowitz, "A Practical Approach to Fractal-Based Image Compression," Proc. Data Compression Conference, Snowbird, UT, pp. 176-185, March 1991.

[26] R. Rinaldo and G. Calvagno, "Image Coding by Block Prediction of Multiresolution Subimages," IEEE Trans. on Image Processing, pp. 909-920, July 1995.

[27] H. Krupnik, D. Malah, and E. Karnin, "Fractal Representation of Images via the Discrete Wavelet Transform," IEEE 18th Conf. of EE, Israel, Tel-Aviv, March 1995.

[28] G.M. Davis, "Self-quantization of Wavelet Subtrees: A Wavelet-Based theory of fractal image compression," Proc. of Data Compression Conference, Snowbird, Utah, pp. 232-241, March 1995.

[29] G.M. Davis, "A Wavelet-Based Analysis of Fractal Image Compression," IEEE Trans. on Image Processing, pp. 141-154, Feb. 1998.

[30] G.M. Davis, "Image Compression via Adaptive Self-quantization of Wavelet Subtrees," Proc. ICASSP-96, Atlanta, USA, pp. 2359-2362, 1996.

[31] A. van de Walle, "Merging Fractal Image Compression and Wavelet Transform Methods," Fractal Image Coding and Analysis: a NATO ASI Series Book, Yuval Fisher, Ed. Springer Verlag, New York, 1996.

[32] J. Li and C.–C. Kuo, "Image Compression with a Hybrid Wavelet-Fractal Coder," IEEE Trans. on Image Processing, pp. 868-874, June 1999.

[33] J. Li and C.–C. Kuo, "Hybrid Wavelet-Fractal Image Compression Based on a Rate-Distortion Criterion," IEEE Conf. on Image Processing, Switzerland, pp. 81-84, Sept. 1996.

[34] L.M. Po and Y. Zhang, "A Novel Subtree Partitioning Algorithm for Wavelet-Based Fractal Image Coding," Proc. ICASSP-98, Vol. 5, 1998.

[35] L.M. Po and Y. Zhang, "Variable Tree Size Fractal Compression For Wavelet Pyramid Image Coding," Signal Processing: Image Communication, 14(3), pp. 195-208, 1999.

[36] T. Kim, R.E. van Dyck and D.J. Miller, "Hybrid Fractal Zerotree Wavelet Image Coding," Signal Processing: Image Communication 17, no. 4, pp. 347-360, April 2002.

[37] L.M. Po, Y. Zhang and Y.L. Yu, "Wavelet Transform Based Variable Tree Size Fractal Video Coding," Proc. ICIP-97 IEEE International Conference on Image Processing, Santa Barbara, California, pp. 294, Oct. 1997.

[38] S.H. Kim and N.C. Kim, "Low Bit-rate Video Coding Using Wavelet-Based Fractal Approximation," IEEE International Conf. on Image Processing, Santa Barbara, USA, pp. 787-790, Oct. 1997.

[39] G.M. Davis and A. Nosratinia, "Wavelet-based Image Coding: An Overview," Applied and Computational Control, Signals, and Circuits, vol. 1, no. 1, 1998.

[40] R.C. Gonzalez and R.E. Woods, "Digital Image Processing," 2nd Edition, Prentice Hall Inc., New Jersey, 2002.

[41] I. Daubechies, "Ten Lectures on Wavelets," Society for Industrial and Mathematics, Philadelphia, PA, 1992.

[42] S.G. Mallat, "Multifrequency Channel Decompositions of Images and Wavelet Models," IEEE Trans. Acoust. Speech Signal Proc., vol. 37, pp. 2091-2110, 1989.

[43] S. Saha and R. Vemuri, "Adaptive Wavelet Filters in Image Coders – How Important are They?," Proc. IEEE/IECON '99, vol. 2, pp. 559-564, Nov 1999.

[44] K. Ramchandran and M. Vetteli, "Best Wavelet Packet Bases in a Rate-Distortion Sense," IEEE Trans. on Image Proc, vol. 2, no. 2, pp. 160-175, April 1993.

[45] B.E.Usevitch, "A Tutorial on Modern Lossy Wavelet Image Compression," IEEE Trans Signal Processing, vol. 18, pp. 22-35, Sept. 2001.

[46] J.M. Shapiro, "Embedded Image Coding using Zerotrees of Wavelet Coefficients," IEEE Trans Signal Processing, vol. 41, pp. 3445-3462, Dec. 1993.

[47] J.M. Shapiro, "A Fast Technique for Identifying Zerotrees in the EZW Algorithm," Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 3, pp. 1455-1458, May 1996.

[48] A. Said and W.A. Pearlman, "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," IEEE Trans Signal Processing, Vol. 6, pp. 243-249, June 1996.

[49] Y. Sun, H. Zhang and G. Hu, "Real-Time Implementation of a New Low-Memory SPIHT Image Coding Algorithm Using DSP Chip," IEEE Trans. on Image Processing, Vol. 11, No. 9, pp. 1112-1116, Sept. 2002.

[50] M.J. Tsai, J. Villasenor and F. Chen, "Stack-Run Image coding," IEEE Transactions on Circuits and Systems for Video technology, vol. 6, pp. 519-521, October 1996.

[51] Z. Xiong, K. Ramchandran and M. T. Orchard, "Space-Frequency Quantization or Wavelet Image Coding," IEEE Trans. Image Processing, 1997.

[52] X. Wu, "High-order Context Modelling and Embedded Conditional Entropy Coding of Wavelet Coefficients for Image Compression," Proc. of 31st Asilomar conf. on Signals, Systems and Computers, pp. 1378-1382, 1997.

[53] X. Wu, "Compression of Wavelet Transform Coefficients, The Transform and Data Compression Handbook," E.K. Rao et al. Boca Raton, CRC Press LLC, 2001.

[54] Image Communications Lab, "Wavelet Image Coding: PSNR Results," UCLA School of Engineering and Applied Sciences. Available http://www.icsl.ucla.edu /~ipl/psnr_results.html.

[55] L.R. Iyer, "Image Compression Using Balanced Multiwavelets," Virginia Polytechnic Institute and State University, June 2001.

[56] P. Xiao, "Image Compression by Wavelet Transform," Department of Computer and Information Sciences, East Tennessee State University, Aug. 2001.

[57] D. Marpe, H. Cycon, "Very Low Bit-Rate Video Coding using Wavelet Based Techniques," IEEE Trans. Circ. And Syst. for Vid. Tech., vol. 9, no. 1, pp. 85-94, Feb. 1999.

[58] I.J. McIntosh, "Implementation of an Application Specific Low Bit-Rate Video Compression Scheme," MScEng Thesis, University of Natal, Feb. 2002.

[59] E.S. Jackson, "High Ratio Wavelet Video Compression through Real-Time Rate-Distortion Estimation," MScEng Thesis, University of Natal, July 2003.

[60] I.H. Witten, R.M. Neal and J.G. Cleary, "Arithmetic Coding for Data Compression," Comm. ACM, 30, no. 6, pp. 520-540, June 1987.

[61] P.G. Howard and J.S. Vitter, "Arithmetic Coding for Data Compression," Proc. of the IEEE, 82(6), pp. 857-865, June 1994.

[62] M. Nelson, "Arithmetic coding + Statistical Modelling," Dr. Dobb's Journal, Feb. 1991. Available at http://www.dogma.net/markn.

[63] R. E. Ziemer and R.L. Peterson, "Introduction to digital communications," MacMillan Publishing Company, Republic of Singapore, 1992.

[64] A. Skodras, C. Christopoulous, and T. Ebrahimi, "The JPEG-2000 still image compression standard," IEEE Signal Processing, vol. 18, pp. 36-58, Sept. 2001.

[65] J. Wiseman, "An Introduction to MPEG Video Compression". Available at: http://members.aol.com/symbangrl.

[66] ITU-T Draft Recommendation, "Video Coding for Low Bit-rate Communication", Dec. 1995.

[67] D. Turaga and T. Chen, "ITU-T Video Coding Standards," Electrical and Computer Engineering, Carnegie Mellon University, 2000.

[68] The Mathworks Inc, "Matlab, The Language of Technical Computing," Version 6.5.0.1 Release 13, June, 2002.

[69] G.K. Wallace, "The JPEG Still Picture Compression Standard," Comm. ACM, vol. 34, pp. 30-44, 1991.

[70] A.N. Skodras, C.A. Christopoulos, T. Ebrahimi, "JPEG2000: The Upcoming Still Image Compression Standard," Pattern Recognition Letters, Vol. 22, pp. 1337-1345, Oct. 2001.

[71] The .TV Company, "Welcome to Motion JPEG2000". Available at: http://www.motionjpeg2000.tv.

[72] R. Koenen, "MPEG-4 – Multimedia for our time," IEEE Spectrum, Vol. 36, pp. 26-33, Feb. 1999.

[73] P. Ogunbona, I. Kharitonenko and P. John, "JPEG 2000 – The New Wave in Image Compression," CCTV magazine, pp. 36-38, Jan. 2000.

[74] Fraunhofer Institut Integrierte Schaltungen, "Audio and Multimedia MPEG-4 Video Coding," Available at http://www.iis.fraunhofer.de/amm/techinf/mpeg4/video.html

[75] M. Gallant, G. Cote, B. Errol, "H.263+ TMN2.0 Reference Software," University of British Colombia, Canada. Available at http://www.ee.ubc.ca/image.

[76] "(MPEG-4) Video Reference Software Version: Microsoft-FPDAM1-1.0-000703 Version 2" ISO/IEC IS 14496-2 PDAM 1.0-000703, July 2000. Available at: http://www.iso.ch/iso/en/ittf/PubliclyAvailableStandards/14496-5_Compressed_directories/Visual/

[77] W. Sweldens, "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions," Wavelet Applications in Signal and Image Processing III, pp. 68-79, 1995.

[78] A. Islam and W.A. Pearlman, "An embedded and Efficient Low-Complexity Hierarchical Image Coder," SPIE Conf. on Visual Communications and Image Processing, San Jose, CA, pp. 294-305, Jan. 1999.

[79] E. Hong, R.E. Ladner and E.A. Riskin, "Group Testing for Image Compression using Alternative Transforms, " Signal Processing: Image Communication, 18 , pp. 561-574, Aug. 2003.

[80] N. Ponomarenko, V. Lukin, K. Egiazarian and J.T. Astola, "Modified Horizontal Vertical Partition Scheme for Fractal Image Compression," Proc. of the 5th Nordic Signal Processing Symposium, Hurtigruten, Norway, Oct. 2002.