Distributed video through telecommunication networks using fractal image compression techniques

Vassilios D. Diakoloukas

DISTRIBUTED VIDEO THROUGH TELECOMMUNICATION NETWORKS USING FRACTAL IMAGE COMPRESSION TECHNIQUES

Vassilios D. Diakoloukas

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE COUNCIL OF THE UNIVERSITY OF DURHAM FOR THE DEGREE OF MASTERS OF SCIENCE (MSc.).

NOVEMBER 1995

Declaration

I hereby declare that this thesis is a record of work undertaken by myself, that it has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged.

V.D.Diakoloukas, November 1995

Abstract

The research presented in this thesis investigates the use of fractal compression techniques for a real time video distribution system. The motivation for this work was that the method has some useful properties which satisfy many requirements for video compression. In addition, as a novel technique, the fractal compression method has a great potential.

In this thesis, we initially develop an understanding of the state of the art in image and video compression and describe the mathematical concepts and basic terminology of the fractal compression algorithm. Several schemes which aim to the improve of the algorithm, for still images are then examined. Amongst these, two novel contributions are described. The first is the partitioning of the image into sections which resulted in significant reduction of the compression time. In the second, the use of the median metric as alternative to the RMS was considered but was not finally adopted, since the RMS proved to be a more efficient measure.

The extension of the fractal compression algorithm from still images to image sequences is then examined and three different schemes to reduce the temporal redundancy of the video compression algorithm are described. The reduction in the execution time of the compression algorithm that can be obtained by the techniques described is significant although real time execution has not yet been achieved.

Finally, the basic concepts of distributed programming and networks, as basic elements of a video distribution system, are presented and the hardware and software components of a fractal video distribution system are described. The implementation of the fractal compression algorithm on a TMS320C40 is also considered for speed benefits and it is found that a relatively large number of processors are needed for real time execution.

Acknowledgments

I would like to thank my supervisors Professor Alan Purvis and Dr. Simon Johnson for their guidance and advise over the last year. Their comments, suggestions and corrections have been extremely helpful.

I am also greatful to my colleague and friend Dionissios Batjakis who sacrificed much of his valuable time helping me with my scientific and language problems. Many thanks to Edy Bertolissi for his advise on distributed programming and networks and to Jiannis Pachos for giving me somewhere to live during the final few weeks.

Finally I would like to acknowledge and give thanks to all the people that made my stay in Durham enjoyable and to my parents who supported me throughout this year.

Table of Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

The ways in which people communicate are changing rapidly. The conventional voice call over a wired network is no longer the only reasonable and reliable method for transmitting information. Instead, the options are many and diverse, ranging from voice calls over wireless networks to video calls over the conventional telephone network, as well as the transmission of arbitrary images or audio signals through a computer network (distributed multi–media systems).

This revolution in communications is being supported by several sources. These include continuously improving network performance, widely available modern workstations with fast processors and high resolution colour graphics displays as well as powerful parallel processors such as DSP systems or transputer networks. While these systems can support computationally demanding applications, their cost continues to fall as the processing power increases.

Based on such a hardware evolution, modern telecommunications technology is progressing rapidly by developing improved algorithms for digital audio and image processing and creating world–wide standards for communication protocols and data handling through the network. All the above contribute to an increase in the use of

modern telecommunications in a wide range of fields and the spreading of this technology in every day life.

1.1. Signal Compression

The rapid increase in the interest of researchers and users, that has recently occurred in the area of storage and fast transmission of multimedia information on a distribution system, is now well known. One of the biggest challenges and an area of intense research related to the construction of such systems, is how to reduce the vast amounts of data and processing time, that are required to represent the multimedia signal of any form (sound, image,video). This area of signal processing is called *signal compression.*

The aim of signal compression is to achieve a low–bit rate in the digital representation of an input signal, with minimum perceived loss of signal quality. A low–bit rate signal decreases the large data storage capacity required and increases the speed of storing and transmission, reducing at the same time the cost. In this project, we intend to experiment with the realisation of a distribution system for image sequences (video frames) and so we will examine image and video compression techniques.

Video signals are very demanding in terms of processing and transmission due to enormous amounts of data that needs to be represented. Normally, a video signal is generated by the collection of a large number of digital image frames. Each of these frames is created by sampling and quantizing a 2–D light intensity function. Supposing that the resulting digital image is 8–bit and of 1024x1024 pixels size, then more than one million bytes are needed to store each frame. Thus, providing adequate storage for a video distribution system is usually a considerable problem.

In addition, communication in video distribution systems often involves local communication between different systems and remote communication from one point to another for the transmission of the whole video data stream. However, communication

across large distances presents an even more serious problem. A typical voice–grade telephone line can transmit at a typical rate of 9600 bits/sec. Thus, in order to transmit a 512x512, 8–bit image at this rate will require nearly five minutes. Although this rate continuously improves with the use of new technologies, such as optical fibres instead of the traditional cables, ISDN and ATM networks, it is still far from a real–time transmission of a video signal. Having the additional disadvantage of the high cost of new technologies, we can conclude that such a process is not yet trivial.

A number of image compression techniques have been developed and used in recent times to address the problem of low cost transmission of video data. These can be classified into two broad categories:

- *Lossless* compression techniques, which allow perfect reconstruction without any loss of detail of the image. These techniques are necessary in applications where the digital image has to be precisely known (as in the storage and transmission of medical images).

- *Lossy* compression techniques, which provide higher levels of data reduction with small loss of the image detail. These techniques are useful in applications where a certain amount of error is an acceptable trade–off for increased compression performance. Such applications are tele–videoconferencing, broadcast television, facsimile transmission systems (FAX) and other multimedia and communication applications.

Some of the most widely used image compression techniques, particularly in video–conferencing systems are H.261 codec, MPEG, M–JPEG (the JPEG analogue for video), the vector–quantization method and wavelet–based encoding. These and other forms of compression will be discussed in more detail in chapter two. A novel lossy compression technique which has become very popular is fractal image compression. In

this thesis we will concentrate on the use of this technique and explore several schemes based on fractals.

1.2. Objectives and motivation for the project

The objective of this project is to investigate the use of fractal image compression techniques for a video distribution system and examine ways in which the algorithm could be improved in terms of speed and image quality, in order to meet the needs of a tele–videoconferencing system. Fractal image compression was chosen instead of other more common image compression techniques due to several advantages.

The first is resolution independence. This property allows the reconstruction of the compressed image to any size since the algorithm is able to create artificial data and avoid the pixelisation phenomenon, common to enlarged digitised images. In addition, fractal image compression can provide very high compression ratios with minimum loss of detail.

Another property is an asymmetry which exists in the time required for the compression and decompression processes. Indeed, decompression can be implemented at very high speed, whereas compression remains slow. This property induces us to examine the use of powerful hardware to speed up the compression process. Since the compression algorithm is based on block partitioning it would seem that it can be readily parallelised, which means that it can be implemented on a parallel hardware consisting of many processors.

Considering the potential advantages of fractal image compression outlined above, it would seem that this method might be particularly appropriate for a videoconferencing system. In this system, the main lecturer will be recorded by a powerful image processing system based on a system of parallel processing units, which will be able to compress video–frames in real–time and at high compression ratios, and then rapidly

transmit them. On the other hand due to the asymmetry mentioned above, the audience should be able to use a general purpose computer to decompress and view the video frames at video rates (25 complete images per second).

1.3. Overview of the Thesis

This thesis has been structured into nine chapters. The first chapter introduces the thesis by presenting the tendency of modern telecommunication technology to transform the traditional telecommunication systems into visual or even multimedia ones. In addition, the problem of the large amount of data needed for the digital representation of images is presented and the consequent need for signal compression is discussed. Finally the background and objectives of the project are explained.

In chapter two, a brief history of image and video compression is given and some of the most commonly used techniques are explained. A comparison of their performance and a description of their advantages and disadvantages is also presented.

Chapter three discusses the basic principles of the fractal compression technique which are used in this project. A flow–chart of the main algorithm is presented and explained in detail. The algorithm in this primitive form has many disadvantages which are discussed, the most important being the time needed for the compression and decompression process. In chapter four we present a number of different techniques which may be used to improve the algorithm and the results of these are discussed.

The implementation of the fractal compression algorithm on image sequences is presented in chapter five. The basic aim is the reduction of the compression and decompression time by reducing the temporal redundancy. Three different schemes are implemented and their performance is compared.

In chapter six, we introduce the network programming that is required for the distribution of video streams. The network philosophy and several network topologies

are described. Furthermore, network protocols, sockets and remote procedure calls are presented.

Chapter seven describes hardware and software that is required in a system for video distribution through the network. The hardware used in the system, the image processor, DCC cameras, workstations and PC's are described. In addition, a TMS320C40 processor is used to speed up the compression algorithm and an estimate of the total number of processors needed for a real time implementation of the fractal image compression algorithm is given. The software required, such as the XWindows protocol and XView toolkit which is needed for the generation of the graphical interface, are also described.

Finally the overall evaluation of the system is concluded.

Chapter 2

STILL AND MOVING IMAGE COMPRESSION

An area in which a lot of on–going research is being done lately is the area of video distribution systems. Video–phone, video–mail and video–conferencing systems have been already developed on industry and academia. All of these systems require that the captured video frames are compressed using a particular compression technique. In this chapter we will describe some fundamentals on the image compression and will present some of the most commonly used compression techniques in order to be able to compare them and evaluate their pros and cons.

2.1. Mathematical representation of an image

Before we further proceed to describe the fundamentals and the state–of–the art on image compression techniques we have to define the image in mathematical terms and view some of its properties.

First of all we will assume that all the images we are going to refer from now on are digital gray–scaled images. Such an image, can be described as a set of points on the space R^2 of the 2–D real world images and is represented by a graph of a function

z=f(x,y). The values of this function are integers which correspond to the gray level of the pixel at position (x,y) and range from 0 up to $b-1$ where b is the total number of possible gray levels. The latter is defined as $b=2^k$ where k is the maximum number of bits that are used to represent the gray level of each pixel. We will normally consider 256 gray levels for 8–bit gray–scaled images.. An example of such graphical representation of the image is shown on figure 2.1.

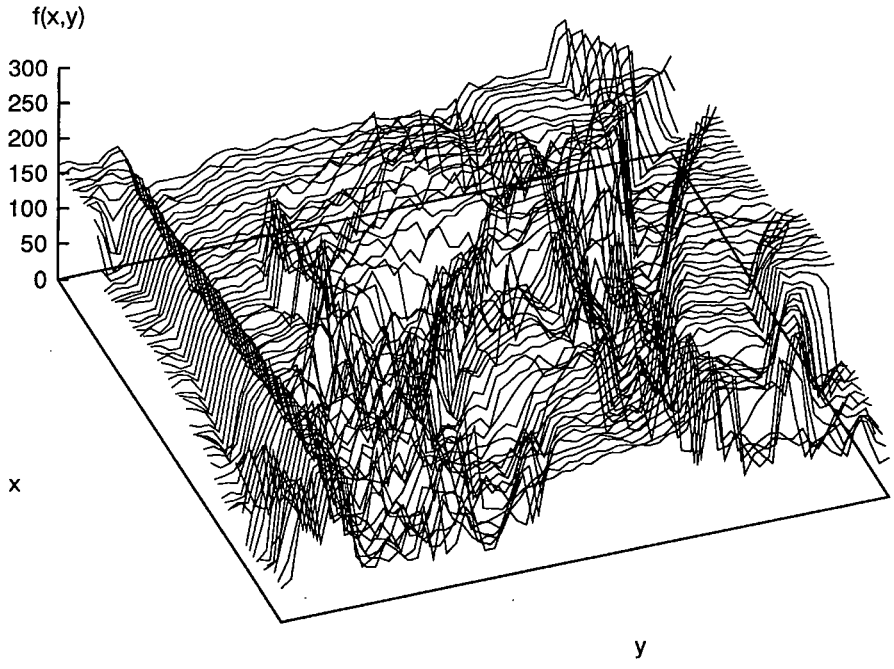


Figure 2.1: Lena image as a 3D graph of the function f(x,y)

2.2. The Need for Compression

The technology in telecommunication networks is developing rapidly providing an increased amount of available bandwidth. However the need for data compression will always remain as the volume of imaging and video data increases exponentially in a very wide variety of applications and as the needs for storing, retrieving and transmitting image and video expands. Somebody could assume that in the future the amount of

bandwidth and the capacity of the storage media will become practically unlimited but this would mean that we are not aware of the situation and the several aspects of the problem and even more we don't learn from the past.

As an example we will refer to the requirement for signal compression due to the transmission of colour images for television using a bandwidth that was originally allocated for monochrome signals. In addition as the technology of High Definition TV (HDTV) comes closer to the public the need for compressing the signal will become a matter of urgency due to the increase of the amount of data that needs to be transmitted.

The usage of the internet and its increased popularity is another example. Although until a few years ago the internet was restricted for the use of the academic institutes and researchers only it now tends to become a network in which everyone in the world would have access. But as its popularity increases so does the demand for immediate access on its databases and the demand for improved multimedia facilities such as on–line sound, graphics, images and full–motion video. Therefore the available bandwidth is never enough. Again the problem has to be addressed by some compression algorithms.

Finally we have to consider the matter of cost. Storing more information will provide additional needs to expensive storing media whereas the transmission of this information will require more time or more bandwidth which can be translated on high payments for the network connection. Therefore even if there will be in the future storing media of big enough capacity and networks which will provide adequate bandwidth the issue of cost will remain and the demand for data compression will be again pressing.

2.3. Some Fundamentals on Image Compression

The aim of the image compression is to encode images or image sequences into as few bits as possible while the original image will be able to be reconstructed with an acceptable visual quality using a decoding mechanism. In the two issues of the image compression that were mentioned (*high compression ratios* and *resulted image quality*) we have to add the issue of *speed* for both the compression and decompression process. The latter is very important for real–time applications such as the video–conferencing systems.

Digital images can be compressed by eliminating the redundant information. There are several types of redundancy that can be exploited by the image compression systems:

- *Interpixel redundancy*: It is the redundancy due to the strong correlation between the adjacent pixels which results from the structural or geometrical relationship between the objects in the natural images. Also referred to as spatial or interframe redundancy. In order to reduce this redundant information the 2–D pixel array is transformed to another more convenient format using several types of transforms (mappings).

- *Coding redundancy*: The spectral values for the same pixel location in images composed from more than one spectral bands are often correlated. This is referred as coding or spectral redundancy. In general coding redundancy is present when the codes that are used to describe some events (i.e. the pixel values) have not been selected to take full advantage of the probabilities of the events.

- *Psychovisual redundancy* : This redundancy refers to the natural property of the human eye to respond with equal sensitivity to all visual information. Certain information has simply less relative importance than other in normal visual processing.

- *Temporal redundancy* : It is due to the fact that adjacent frames in a video sequence often show very small difference. This type of redundancy is removed by techniques that only encode the differences between adjacent frames in the sequence, such as motion prediction and compensation.

As mentioned in the introduction of the thesis all image and video compression methods can be divided into two main categories, these being the *lossless* or *reversable* and the *lossy* or *irreversable.* Lossless compression is adequate when low compression ratios are acceptable or in applications where even little loss of detail is not acceptable as in medical imaging or military inspection. On the other hand higher compression ratios can be achieved by lossy compression schemes. In most of the applications the lossy compression will be acceptable as long as the quality of the resulting image is good enough. Since in a video conferencing system as the one proposed here there is usually no need for the video signal to be perfectly reconstructed the lossy compression techniques will be more appropriate.

2.3.1. Compression Performance Measures

Although the evaluation of the quality of the reconstructed images is something subjective depending on the eye perception and other certain factors we will however try to define some objective measures that will be used to lead the reader throughout this thesis. The first measure, the *compression ratio* is defined as:

$$C_R = \frac{n_1}{n_2}$$

where n_1 denotes the number of bits in the original image and n_2 the number of bits in the compressed image. In addition the measure of the *peak signal to noise ratio* is defined as:

$$PSNR = 20\ log_{10}\left(\frac{b-1}{RMS}\right)$$

where b denotes again the number of possible gray–levels (typically 256) and RMS is the Root Mean Square Error defined as:

$$RMS = \sqrt{\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}(f(i,j) - g(i,j))^2}$$

where N and M denote respectively the width and height of the images in pixels, f is the function which represents the original image and g the function of the reconstructed image. These two images must be of the same size.

2.4. State–of–the Art in Still and Moving Image Compression

The subject of digital image compression, often referred to as image coding, has been addressed by many researchers. Several methods have been developed such as statistical methods and transform coding methods. Most of these methods originate in the field of traditional digital signal processing and aim to reduce the different types of digital image redundancy (interpixel, coding and psychovisual redundancy) by combining several techniques. Some other methods that have been investigated are based on the construction of models of the image data. In the following sections a brief description of the most commonly used and investigated compression techniques will be given.

2.4.1. JPEG and M–JPEG compression

A few years ago a joint of the ISO and CCITT committies known as JPEG (Joint Photographic Experts Group) worked and succeeded to establish the first international compression standards for continuous–tone still images both full–color and gray–scale. These standards combine several kinds of primitive coding techniques such as the transform coding, Huffman coding, vector quantization, run–length coding and others in order to obtain better coding efficiency[13].

The basic aim of the JPEG proposed standard was to be able to support a wide variety of application. In order to meet the differing needs of the numerous applications the algorithm includes four modes of operation these being the following [7]:

- *Lossless encoding.* Although JPEG was proposed as a lossy compression technique there is an option based on a predictive method, to operate image encoding which guarantees its exact recovery. However in that case the result is low compression ration compared with the lossy modes. This encoding mode is the only one which is not based on the Discrete Cosine Transform (DCT).

- *Hierarchical encoding.* The image is encoded at multiple resolutions so that lower–resolution versions may be accessed without first having to decompress the image at its full resolution.

- *Progressive encoding.* In this mode the image is encoded in multiple scans for real time transmission applications (the viewer can watch the image to be constructed in multiple coarse to clear passes.)

- *Sequential encoding.* Each image component is encoded in a single left–to–right top–to–bottom scan. This encoding mode is the most widely used for the compression of real world scenes.

Generally speaking the JPEG algorithm achieves much of its compression by exploiting known limitations of the human eye (For instance small colour details of light and dark that are not perceived).

The key processing steps which are the heart of the DCT–based modes of operation are shown in figures 2.2 and 2.3. These figures illustrate the special case of single–component (grayscale) image compression. In order to understand the essentials of the DCT based compression we have to think of it as compression of a stream of 8x8

blocks of grayscale image samples. In the case of colour images the image compression can be regarded as compression of multiple grayscale images.

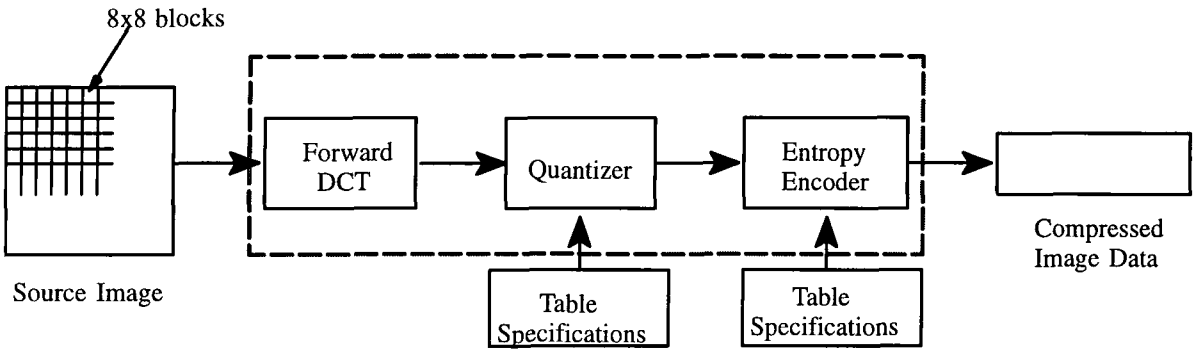


Figure 2.2: *DCT–based encoder [7]*

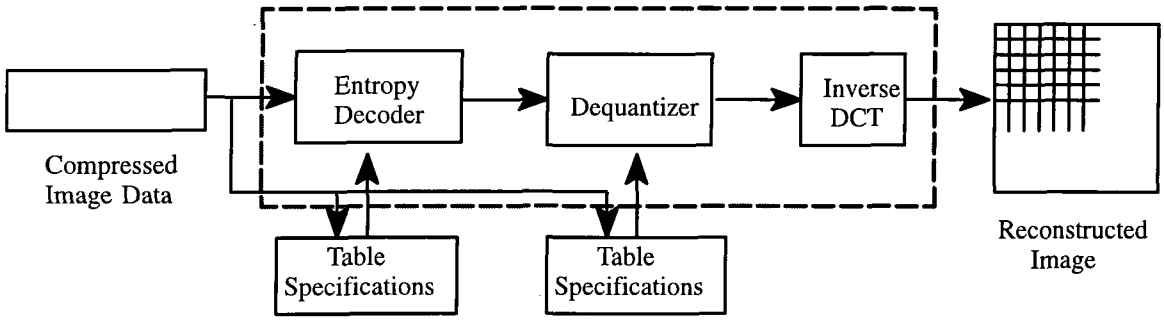


Figure 2.3: *DCT–based decoder [7]*

Each of the 8x8 blocks serves as input and makes its way through the processing steps to yield compressed output into the data stream.

The encoding process involves three successive operations. These are the mapping the quantization and the entropy coding. The mapping operation transforms the image into an array of transform coefficients in order to de–correlate the image data. This reduces the interpixel redundancy and the quantizer and coder in the next stage can be used more efficiently. The decorrelating transform that has been used is the Discrete Cosine Transform DCT which has been found to have computational advantages over

other possible transforms (i.e. Discrete Fourier Transform, Discrete Hartley Transform e.t.c.).

In the next step, the quantizer divides each DCT coefficient by a step size to reduce the accuracy of the mapper's output. This irreversible operation eliminates the psychovisual redundancy of the input image.

In the final encoder processing step, the entropy coding, additional lossless compression is achieved by encoding the quantized DCT coefficients more compactly based on their statistical characteristics. The entropy coding methods that are used for JPEG compression are the Huffman coding and arithmetic coding.

Although JPEG compression was originally meant to address still–frame images, it has been found to perform quit well in motion sequence compression and decompression for multimedia, video editing and remote sensing applications. This technology has come to be known as Motion JPEG or in short M–JPEG and compresses every frame individually using the standard JPEG algorithm for high image quality.

With its frame approach to sequence compression no images are lost and therefore may be manipulated, analysed, archived and viewed discretely. On the other hand the algorithm is computationally intensive thus special hardware is needed for its implementation. However it seems to grow into a fairly good business for some hardware integrators since it affords reasonable cost authoring on the desktop.

2.4.2. MPEG compression

For the establishment of a world standard for coded representation of moving pictures another group the Moving Picture Experts Group (MPEG) was created and worked under the joint supervision of the ISO and IEC [15]. The basic aim of the group was not to specify a particular encoding process but instead to define a compressed bit stream which will implicitly define a corresponding decompressor. Therefore the com-

pression algorithms are up to the individual manufacturers and this is where proprietary advantage is obtained within the scope of a publicly available standard.

Initially the MPEG committee wanted to create a special standard for the delivery, storage and retrieval of video and audio on CD–ROM's[1]. The targeted bit rates for the video signal were around 1.2 Mbits per second whereas for the stereo audio signal was around 250 Kbits/sec. This target was achieved by the MPEG–1 standard which defines a bit stream for compressed video and audio optimized to fit into a bandwidth of 1.5 Mbits/sec.

Distribution network industries, such as cable television and narrow channel satellite broadcasting realizing the potential of digital compression technology to increase services and lower costs, liked the MPEG concept but they were not limited by CD–ROM data rates. Concequently, MPEG developed a second effort that takes advantage of the higher bandwidths available to these networks to deliver higher image resolution and picture quality. This second effort targeted to increased image quality in ranges from about 3 to 15 Mbps, support of interlaced video formats, and provision for multi–resolution scalability. The standard that was developed by this effort is known as MPEG–2.

Additional efforts have been made to further improve the efficiency of these standards and creating others such as the MPEG–3 and MPEG–4. In particular MPEG–3 was going to be a higher quality encoding for High Definition Television (HDTV). However the effort was abandoned after some studies proved that MPEG–2 at higher rates is pretty good. On the other hand MPEG–4 aims at the opposite extreme; that of low bandwidth or low storage capacity environments.

The MPEG standards extend the DCT–based compression approach described in the preceding section to include methods for reducing frame–to–frame redundancies[14].

1. Compact Disk Read Only Memory

The basic scheme is to predict motion from frame to frame in the temporal direction and then to use DCT's to reduce the redundancy in the spatial direction. In other words, starting with a relatively low resolution video sequence (possibly decimated from the original) the MPEG algorithm compresses a starting reference frame using JPEG–like DCT–based approach, then it reconstructs the compressed frame and estimates the motion of objects between the reconstructed frame and the next frame. Based on the amount of motion it decides after that whether to compress the next frame independently or by using references to the previously coded frame.

The motion estimation step normally involves the comparison of each reconstructed sub–image with every immediate neighbourhood of it, in the next or previous frame (there are backward prediction modes where later frames are sent first to allow the interpolation between frames). This means that a measure of correlation such as the sum of the square of the pixel–by–pixel differences is computed in order to find a close match.

After the DCT is applied, the produced coefficients are divided by some value to drop bits out (quantization). During this process hopefully many of the coefficients will become zero reducing the information that needs to be saved. The resulting information from all these steps which include the DCT coefficients the motion vectors and the quantization parameters are Huffman coded using fixed tables.

In practice the scheme that we described is more complicated. There are three types of coded frames. The first one is the intra frames which are coded as they were still images. Another type is the predicted frames which are predicted by the most recently reconstructed intra or predictive frames. Finally there are bidirectional frames which are predicted directly by two frames, the previous and the next one. Because the MPEG standard is intended for applications in which many rapid scene changes may occur, it

specifically requires that every 15th frame be encoded without reference to any preceding frames. This requirement is also helpful in video editing applications.

In general, MPEG is an asymmetric compression technique. The overall compression process described above can be computationally very intensive whereas the decompression process normally requires considerably smaller time to be implemented.

2.4.3. H.261 Codec

Unlike the MPEG standards which were designed to cover the growing need of generic coding methods for moving images in a wider scope of applications the H.261 standard was specifically designed to meet the needs for the compression of moving images in video conferencing systems. The standard describes the video coding and decoding methods for the moving picture component of an audiovisual service at rates of p*64 Kbps where p is in the range of 1 to 30. It basically aims and is really suitable for applications using circuit switched networks as their transmission channels.

The H.261 standard is similar to the MPEG DCT–based compression mechanism but differs in the manner in which motion estimation is handled. In the H.261 each frame is compared to a single preceding frame. The encoder consists of three main operations these being the prediction the block transform and the quantization. The prediction operation is used for the inter coding frames only which are encoded with respect to another reference frame using a prediction error calculated between the corresponding 16x16 pixel regions of two successive frames. The intra coding frames in which blocks of 8x8 pixels are encoded with respect to themselves are sent directly to the block transformation process.

Similarly with the MPEG standard, in this step each block is processed by a forward DCT function and then the produced DCT coefficients will be quantized in order to represent them more coarsely. Further compression is achieved by applying Huffman coding as the entropy coding technique.

H.261 is usually used in conjunction with other control, multiplexing and framing standards such as H.221, H.230, H.242 and H.320. It is a compression standard commonly implemented in hardware to reduce its computational complexity. PC cards of this kind for video, audio and ISDN do exist.

2.4.4. Compression Techniques Under Investigation

Furthermore to the compression standards and techniques that were described in the previous sections, some other less popular ones have been used in many applications such as Intel's Digital Video Interactive (DVI) scheme and Apple's Quicktime and Microsoft's Video for Windows standards which do not specify a particular video format but instead a framework to accommodate many video formats. Furthermore researchers continue to investigate several other compression techniques. We will shortly describe some of them:

- A novel approach on Image compressors / decompressors is based on the wavelets transform[9][11][17]. The research on the wavelets has made a tremendous progress in the last few years resulting on symmetrical image compression schemes with many advantages. For example these schemes seems to overcome the capabilities of the JPEG in terms of quality of the reconstructed image specially on large compression ratios. However the transformation (forward and reverse) is computationally very expensive this being its major disadvantage. Some experiments that have been made to compress image sequences were based on the compression of the coarse image of the difference between two consecutive frames but even though it still is far from a real–time encoding or decoding.

- Neural Network Image Compression. Several kinds of neural networks have been used to provide image compression such as feed–forward or random neural networks[18][19]. In some cases the neural network was used to find the coefficients of a transformation (2–D Gabor transform). However in most cases in order to perform

image compression the image is first divided into 8x8 blocks and each block is scaled to serve as an input to the network. The network is then trained to produce the same output as seen in the input. Because of the less neurons in the hidden layer than in the input and output layer the image is finally compressed. For the compression of image sequences a motion detection technique is applied by comparing the blocks from two successive frames. The algorithm has the main advantage of being fast and in some cases the compression and decompression process can be carried out in real–time. However the resulted compression ratios are far smaller from the ones in more conventional methods.

- Another common image compression method is the vector quantization[16]. The vector quantizer is in practise a mapper which maps a stream of analogue or very high rate discrete data into a sequence of low rate data. This methods can achieve high compression ratios but suffer from edge degradation and high computational complexity.

Through the bibliography some other compression techniques can be seen some of them based on the Gabor transformation [12] or the 3–D segmentation of sequences. However the most promising recent compression schemes are the ones based on fractals. These techniques consider the existence of a self–similar structure on natural images which enables them to be modelled by a relatively simple mathematical model. They are based on the measure of deviation between a given image and its approximation which is constructed as attractor by an IFS (Iterated Function System) code. Details on the methodology and the basic principles of the fractal image compression process will be described in the next chapter.

2.5. Summary and Conclusions

The basic conclusion that can be extracted is that there is a plethora of activities in the area of image and video compression nowadays. Many different approaches and many different standards compose the total effort for the provision of compression which will combine large compression ratios, high–quality of the reconstructed image as well as fast implementation. However none of the compression techniques combine efficiently all these three aspects of the image compression.

In practice it seems that most people converge to the usage of the well known and internationally established compression standards such as the JPEG, H.261 and MPEG. Although these standards provide a good solution to the problem of the small network band–width they still have some disadvantages including the fact that the encoding process is computationally intensive, the image quality is very poor at high compression ratios, they are not resolution independent and that normally there is a trade off between the degree of compression, the quality of the resultant image and the time needed to be implemented.

Amongst the other compression methods which are under continuous investigation, most of them provide superior performance from the establish standards at some image compression issues but they are inferior to some others. In this project we have chosen to examine fractal based compression techniques.

These techniques for still images gained a rapidly increased interest in the last few years due to the numerous advantages they provide. Some of the strengths of the fractal techniques is that they provide very high compression ratios with a little loss in detail, that the images can be recovered in any resolution and relatively fast; they are also ideally suited to real world image compression, and the compression ratio does not increase linearly with the original image size. On the other hand the major disadvan-

tage is that the compression times are considerably long unless powerful hardware assists the execution of the process.

Because of the later weakness of the algorithm only very few efforts have been made to apply fractal compression on distributed systems. However as the Fractal compression is a recently discovered method and has already been patented much information on how it works is hidden and many techniques are expected to be developed in the near future to improve its performance. In the following chapters we will describe in detail the fractal compression algorithm and several improvements in order to make it capable for video compression.

Chapter 3

FRACTAL IMAGE COMPRESSION

As mentioned in the previous chapter fractal compression of digital images has recently attracted much attention. It is a very promising technique in terms of compression ratios, providing in addition some other advantages such as the resolution independence of the reconstructed image and the asymmetry of the compression / decompression time needed. The purpose of this chapter is to present a concise description of the theoretical background of the fractal image compression technique and introduce the terminology and other requirements for a better understanding of the encoding and decoding implementations.

3.1. Fractals and Geometry in Nature

A fractal is a mathematical object, a chaotic fractured structure, possessing similar forms at various sizes (self–similar). In other words fractals are geometric patterns generated from simple formulas by starting from an initial arbitrary shape and infinitely adding new shapes created by repeating simple transformations such as shrink, rotate, move and flip, until it grows larger than a certain boundary. Fractals are infinitely complex, the closer you look the more detail you see.

Fractals were not discovered in a single instant, but knowledge on them grew quickly in the computer age. The first real fractals were discovered by a French mathematician named Gaston Julia. In his time there were no computers, so serious study of fractal objects was impractical. His studies were later re–discovered by the mathematician Benoit Mandelbrot who was the first that introduced the term fractal. He was the one that furthermore observed the existence of a geometry in nature [3].

Indeed many naturally occurring objects exhibit at smaller scales of detail this self–similarity found in fractals. For instance a tree has large limbs branching out in all directions and each of these limbs has smaller limbs branching out in a similar way. Through experimentation, certain fractal codes have been developed which can successfully model landscapes or other natural effects such as clouds, snowflakes, mountains and forests [4]. Fractal objects produced in this way have the intrinsic property of having extremely high visual complexity while being very low in information content as they can be described and generated by these algorithms.

The later property became the basis for the development of fractal image compression. In order to introduce some basic terminology and help the non–specialist to understand more clearly the way fractal image compression works we will demonstrate some simple forms of fractal objects such as the Sierpinski triangle.

3.2. Attractors and affine transformations

The Sierpinski triangle can be generated from an arbitrary image, by reducing the size of this input image by a half and reproducing it three times at different locations to form the output image as shown in figure 3.1. If this process is repeated indefinitely, feeding back each time as the next input image the output of the previous step, than a convergence to the same final image is observed. This final image is known as ***attractor***

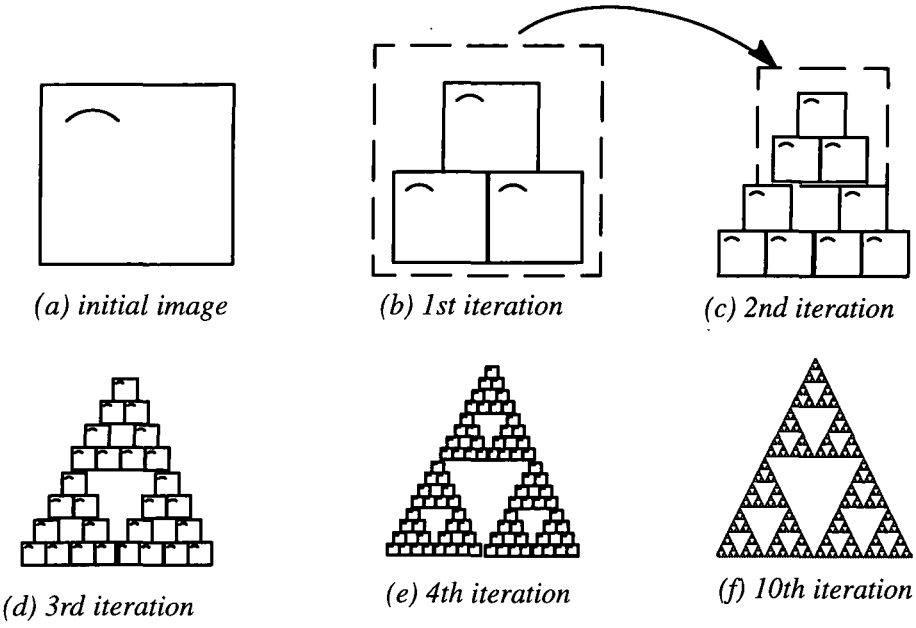


Figure 3.1: *The Sierpinski triangle*

and has the important property of not being affected from the initial image thus being unique for every different set of transformations applied.

The only limitation we have to set is that the transformations must be ***contractive***. A transformation is said to be contractive if and only if, applied to any two points of a given image forces them to get closer to each other. In more mathematical terms the contractivity theorem says that:

$$d(W(f), W(g)) \leq s\, d(f,g),\ 0 \leq s < 1\ ,\ \forall f,g \in R^2$$

where W denotes the applied transformation and f,g are functions representing images of the real world and R^2 denotes the image space. The contractivity condition comes very naturally since in any other case the attractor would have been of infinite size.

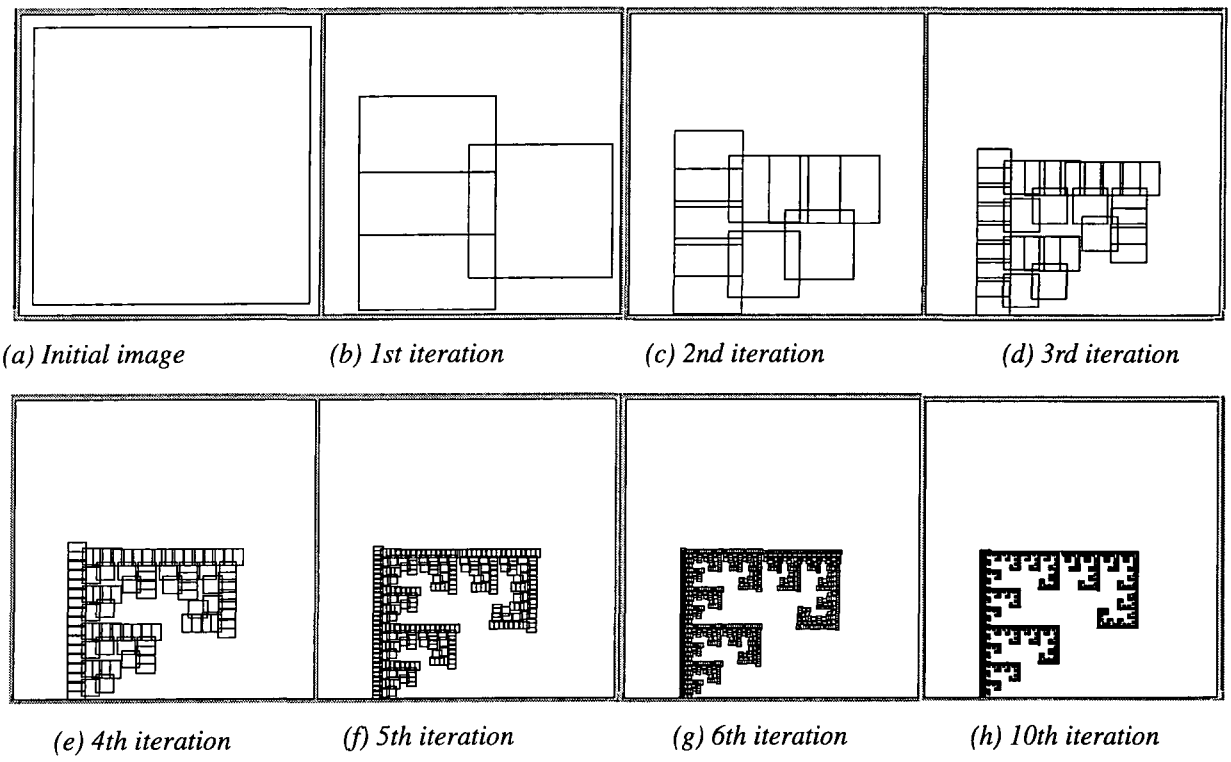
Originally it was thought that in order to ensure that in order to ensure contractivity in the intensity direction it was required that |s|<1 for every range domain mapping. However it was found later [26] that this requirement is far too strong. In practice we can only consider the average of the s values to be less than 1 and therefore we are

allowed to choose some of them being greater than 1 without necessarily creating instability because the transformation will be expansive. In most cases a choice of |s|<1.5 gives the best performance in terms of finding good mappings while the transformations are still contractive.

Although the transformations can have any form, in practice we deal with those having the form:

$$w_i\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}a_i & b_i\\c_i & d_i\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} + \begin{bmatrix}e_i\\f_i\end{bmatrix} \qquad (3.1)$$

which are sufficient for handling an input image in many different ways (i.e. rotate, scale, move). These transformations are known as ***affine transformations***.



(a) Initial image *(b) 1st iteration* *(c) 2nd iteration* *(d) 3rd iteration*

(e) 4th iteration *(f) 5th iteration* *(g) 6th iteration* *(h) 10th iteration*

Figure 3.2: *Reconstruction of a fractal object (attractor) by applying iterativelyfractal transformations*

a	b	c	d	e	f
0.00	0.53	–0.48	0.00	118.00	150.0
0.50	0.00	0.00	0.50	24.00	0.0
0.50	0.00	0.00	0.50	24.00	64.0

Table 3.1: *IFS codes for the attractor of figure 3.2*

In the case of the Sierpinski triangle described above supposing that the size of the image was 256x256 pixels, then the following three affine transformation were used:

$$w_1\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}0.5 & 0\\0 & 0.5\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix} \qquad (3.2)$$

$$w_2\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}0.5 & 0\\0 & 0.5\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} + \begin{bmatrix}128\\0\end{bmatrix} \qquad (3.3)$$

$$w_3\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}0.5 & 0\\0 & 0.5\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} + \begin{bmatrix}64\\128\end{bmatrix} \qquad (3.4)$$

If we change at least one of the parameters to any of the affine transformations then a different attractor will result. The second example shown in figure 3.2 was created by three transformations using the parameters of table 3.1. The union of the contractive transformations, w_i is referred to as an ***Iterated Function System (IFS)*** code. If for instance A is a set representing an input image then the IFS code is defined as:

$$W(A) = \bigcup_{i=1}^{n} w_i(A) \qquad (3.5)$$

The contraction mapping theorem states that any given IFS code consisting of contractive affine transformations can be used to describe a unique and independent from the initial image attractor. This means that if we let:

$$x_W = \lim_{n\to\infty} W^{on}(A) \qquad (3.6)$$

represent the attractor of an IFS code W then:

$$W(x_W) = x_W \qquad (3.7)$$

Attractors formed in such way have detail in every scale therefore they are considered to be forms of fractals.

The use of the process described above, for image compression, was first proposed by M.Barnsley. He suggested that storing natural images as collections of transformations could lead to large image compression. For instance for the image of the attractor in figure 3.2.(e) only the six parameters for each transformation presented in table 3.1 have to be stored in order to be correctly reproduced.

Similarly in the case of any arbitrary image, this could be again stored compactly if a relatively small number of transformations capable to recreate the original image could be found. The main questions that remain however is if a typical image exceeds the self–similarity of the fractals thus containing affine transformations of itself and how these can be found.

In first view such an image does not seem to be self–similar but in practise it contains a different kind of self–similarity. As shown in figure 3.3 sample regions of the image are similar with others in different scales. Thus, while the fractal objects described above were formed with copies of their whole selfs, a natural image can be formed as properly transformed parts of itself. Barnsley developed a method based on the previous notation, called the Fractal Transform which provides an automated way for calculating a set of IFS coefficients that define a given digital image.

3.3. The Collage Theorem

The fractal transform is based on the *collage theorem.* The latter states that the union of contractive affine transformations on an image space defines a map (or IFS code) W on the space. Given an input set (or image) A, a new set or attractor $W(A)$ is described

Figure 3.3: *Self–similarity on a natural image*

by the union (collage) of sub–images, each of which is formed by applying a contractive affine transformation w_i on A. This theorem is used to define the IFS code for the examples of figures 3.1 and 3.2 as the union of the transformations w_1, w_2 and w_3.

The collage theorem is also important because it suggests a method of simplifying an image before determining its IFS code. The collage theorem describes an image attractor as the union of one or more sub–images, each of which is formed by applying a contractive affine transformation on an arbitrary image. Therefore when a given digital image is modelled as an attractor, it may be partitioned into sub–images (referred as *range blocks*), each of which can also be treated as an attractor with its own set of contractive transformations that comprise a local iterated function system or local IFS. The advantage produced by this partitioning is that the smaller range blocks are less complex and can therefore be described by a simpler local IFS. The union of these simplified local IFS codes then forms the original image's Partitioned Iterated Function System or PIFS.

As stated, the target sub–images are referred to as range blocks. Each range block is an attractor that results when a given set of local IFS transformations is applied repeatedly to an arbitrary sub–image of specific size and location. The union of those range blocks form the original image. The source sub–images to which the transformations are applied to form range blocks are referred to as domain blocks. The only restriction in the choice of domain blocks is that these have to be larger than the range blocks it models so that the local IFS transformation will be contractive.

3.4. A Note on Metrics

An important factor in the implementation of the fractal code algorithm is the choice of the metric space and the metric itself. Generally speaking a metric space is a set X of points on which a real–valued function $d : X \times X \rightarrow \mathbb{R}$ is defined, called a metric. The latter function takes two elements of the set X and measures the distance between them. The metric obeys the following axioms:

a) $d(x,y) = d(y,x), \quad \forall x,y \in X$

b) $d(x,y) = 0$, if and only if $x=y, \quad \forall x,y \in X$

c) $0 < d(x,y) < \infty, \quad \forall x,y \in X$ and $x \neq y$

d) $d(x,y) \leq d(x,z) + d(z,y), \quad \forall x,y,z \in X$

The metric set described can be a set of points as well as a set of images. Suppose that we deal with a set of images containing for example all the subsets of the plane. In general the axioms above permit the definition of more than one metric for a given space. Some examples of metrics in the plane are the following [21][22][5]:

- The standard Euclidean metric which gives the distance between the points (x_1, y_1) and (x_2, y_2):

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- Another metric between these points is:

$$d((x_1, y_1), (x_2, y_2)) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$$

- The Euclidian plane with the Manhattan metric.

$$d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$$

For the functions $f,g : \mathbb{R}^2 \to \mathbb{R}$, which describe natural images, the following metrics can be defined:

- The supremum metric which finds the position (x,y) where two images f and g differ the most and sets this value as the distance between f and g :

$$d_{sup}(f, g) = \sup_{x \in \mathbb{R}^2} | f(x) - g(x) |$$

- The RMS (root mean square) metric:

$$d_{rms}(f, g) = \sqrt{\int_{I^2} (f(x, y) - g(x, y))^2}$$

The choice of metric is very important because it determines whether a transformation is contractive or not. In a general metric space the continuously applied transformations are not necessarily convergent [20]. As stated from the Banach's fixed point theorem, in order to ensure that each contractive mapping is associated with a unique fixed point (attractor) the metric has to be *complete*. The later means that every Cauchy sequence of points from the metric space converges to a point of the same metric space [20]. A sequence of points in a metric space is said to be a Cauchy sequence if for any $\varepsilon > 0$ there exists an integer N such that:

$$d(x_m, x_n) < \varepsilon \quad \forall n, m > N$$

The metric spaces of the natural image functions that we described above are all complete therefore they can be used for the implementation of the fractal compression algorithm [22][20]. However the RMS metric will be used since it is more convenient in practice as we will see [21].

3.5. Description of the algorithm

3.5.1. Compression

After explaining the terminology and all the principals of fractal image compression we will now present step by step the algorithm. We will follow in general the algorithm presented by Fisher in [21] and by Barnsley in [22], both implementing the forward and inverse fractal transform. A flow diagram for the compression is shown in figure 3.6.

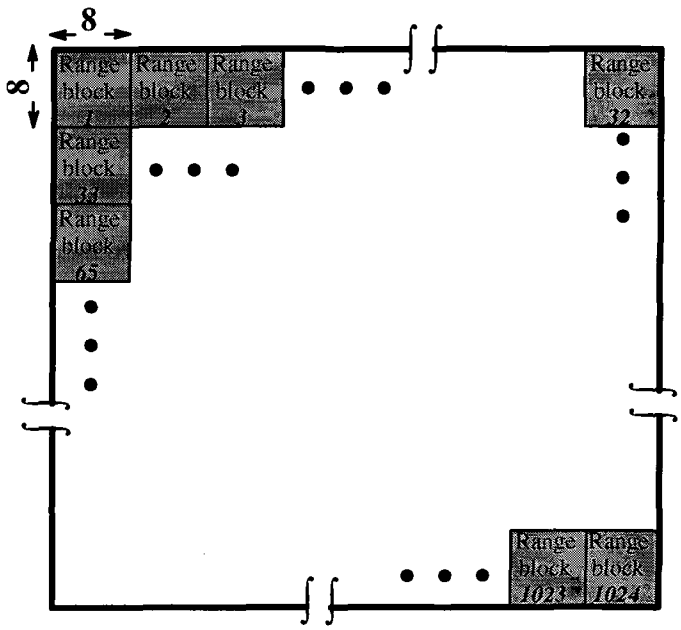


Figure 3.4: *Range block partitioning*

Suppose we have to compress a gray–scaled image of size 256x256 pixels. The image is initially partitioned into a several number of small non–overlapping portions

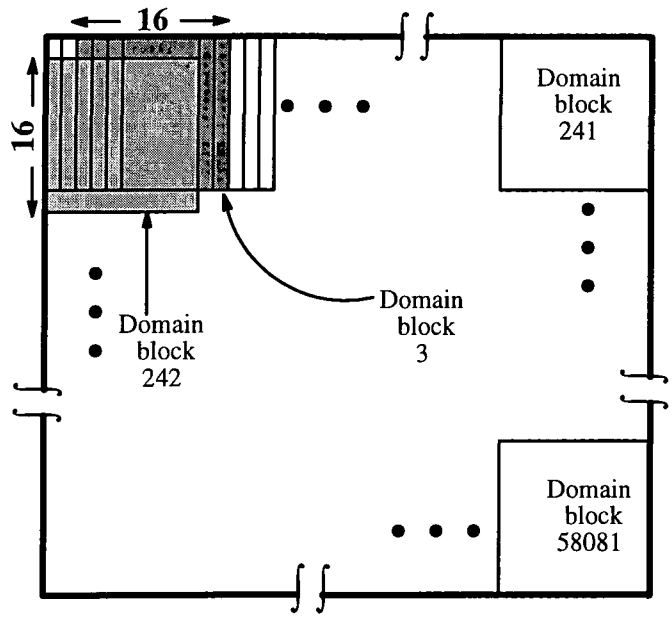


Figure 3.5: *Domain block partitioning*

which serve as the range blocks (i.e. 1024 range blocks each of size 8x8 pixels) as shown in figure 3.4. In addition the image is partitioned into overlapping domain blocks double in size (i.e. 16x16 pixels) which cover every possible region of the image as shown in figure 3.5. Thus the total number of domain blocks reaches 58081. Now every range block can be compared with all the sub–images of the proper size to determine an acceptable match.

We have to note here that in practice the possible comparisons for each range block are much more than the total number of domain blocks referred above due to the existence of eight different ways for mapping one square onto another. These different ways are referred to as isometric transformations or symmetries and are analytically described on table 3.2. The existence of these symmetries rises the total number of all possible comparisons to 8x58081=464648 squares for each one of the 1024 range squares.

Symmetry No	Description	Graph
1	Identity	
2	Reflection in y–axis	
3	Reflection in x–axis	
4	180^0 rotation	
5	Reflection in line x=y	
6	90^0 rotation	
7	270^0 rotation	
8	Reflection in line x=–y	

Table 3.2: *The eight isometric transformations*

Before we proceed with the comparisons a method for mapping a Domain block into a range block has to be introduced. Previously we explained that an affine transformation on the plane had a form of equation (1) where the coefficients a_i , b_i , c_i , d_i determine scale rotation and skew and e_i ,f_i determine translation. However this equation would prove appropriate only for binary images (black and white). Since we are dealing with gray scaled images the pixel intensities have to be transformed in addition in order to match domains with ranges. The latter can be accomplished by representing the image in the three dimensional space where the third axis will repre-

sent the intensity. The matrix of transformation is then of the form:

$$w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \qquad (3.8)$$

where s_i controls the contrast and o_i the brightness. By using equation (3.8) we ensure that the x–y spatial data is not influenced from the intensity data although the later is incorporated as a spatial dimension, since s_i is multiplied only to z, and o_i is added only to z.

After the definition of the transformation in equation (3.8) we are able to compare a domain block with a range block by mapping their geometries and pixel intensities. One of the metrics defined in the previous section have to be used. in order to measure the distance between the two sub–images. One of the most convenient to use metric is one similar to the RMS metric between a scaled by one–half domain block, possessing pixel intensities D_1, ..., D_n, and a range block on its normal size possessing pixel intensities R_1, ..., R_n where n is the total number of pixels for both blocks (i.e. 8x8=64 pixels) defined as[21]:

$$R = \sum_{j=1}^{n} \left[(s\, D_j + o) - R_j \right]^2 \qquad (3.9)$$

where R is RMS^2. In order to calculate this quantity the variables s and o have to be determined. This can be done by taking the partial derivatives of the quantity with respect of the two variables and setting them equal to zero in order to minimize the distance. This produces the following relations:

To avoid redundant computations all the different sums, $\sum_{j=1}^{n} R_j$, $\sum_{j=1}^{n} D_j$, $\sum_{j=1}^{n} D_j^2$ are

calculated once as each new range and decimated domain block is extracted. The

$$s = \frac{n^2\left[\sum_{j=1}^{n} D_j R_j\right] - \left[\sum_{j=1}^{n} D_j\right]\left[\sum_{j=1}^{n} R_j\right]}{n^2 \sum_{j=1}^{n} D_j^2 - \left[\sum_{j=1}^{n} D_j\right]^2} \qquad (3.10)$$

$$o = \frac{\sum_{j=1}^{n} R_j - s \sum_{j=1}^{n} D_j}{n^2} \qquad (3.11)$$

distance R can then be written on the following more computationally efficient from :

$$R = \frac{\sum_{j=1}^{n} R_j^2 + s\left[s \sum_{j=1}^{n} D_j^2 - 2 \sum_{j=1}^{n} D_j R_j + 2\, o \sum_{j=1}^{n} D_j\right] + o\left[o\, n^2 - 2 \sum_{j=1}^{n} R_j\right]}{n^2}$$

Once a range block has been extracted the first pre–computed decimated domain block is taken from the domain pool for comparison. As we notice for each comparison between a domain and a range block only the sum $\sum_{j=1}^{n} D_j R_j$ has to be computed. The process of extracting a different domain block and comparing it with the range block is repeated until one of the comparisons result in a mean square distance less than a specified tolerance. The information describing the local IFS transformations that successfully mapped the domain block to the given range block is than stored as an affine map. In practice these maps consist of some parameters defining the orientation, size and the position of the domain block as well as the scale s and offset o parameters. If no match is found then the affine map of the domain block that most closely matched the given range block is stored.

The whole process is repeated until we find one affine map for every range block of the image, constructing in such way the PIFS information for the image.

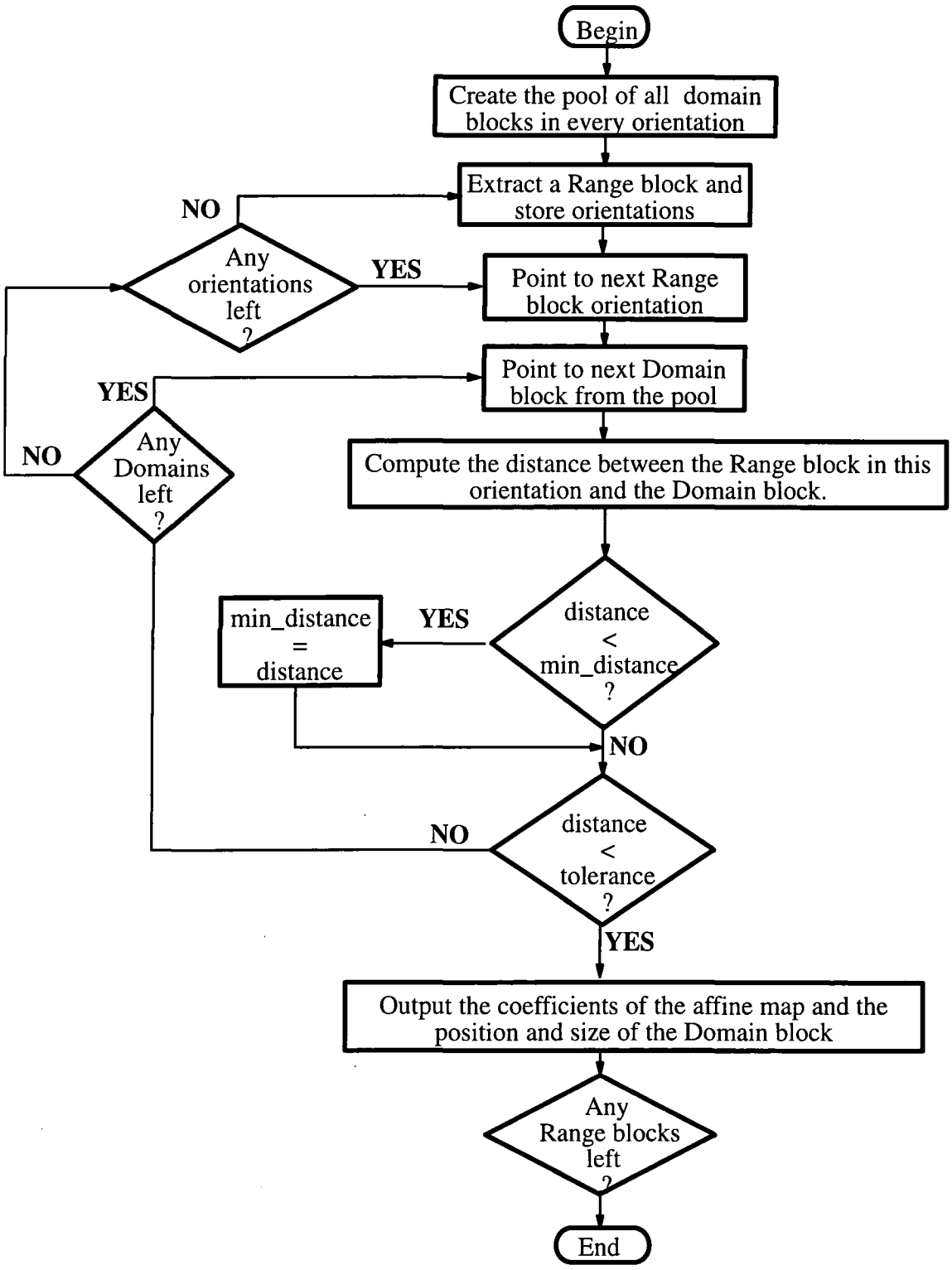


Figure 3.6: *Flow chart of the basic compression algorithm*

3.5.2. Decompression

The decompression is a much simpler process. An arbitrary image serves initially as an input (normally a black box) and is tiled in the same way as in the compression process to define non–overlapping sub–images in the size of the range block. For each of these range regions the corresponding position and size of the domain block and the coefficients of the affine map is read. Then the specified domain block is extracted from the initial image and mapped into the associate range region. In such way the homogeneous black background in the range region is now replaced by a completely new image. The process will continue until the whole image is created. Afterwards the newly created image will serve as input and the whole process will be repeated for a few iterations.

3.6. Summary

In this chapter we introduced the basic concepts of the fractal image compression technique. The terminology was defined and the mathematical background was explained. The fractal block coding is based on the collage theorem and the contraction mapping theorem. Another important aspect of the fractal compression technique is the definition of an appropriate metric space. The RMS metric was analysed in more detail since this was found to have many advantages such as the easy calculation of the scaling and offset parameters. Finally the description of the basic form of both the compression and decompression algorithms was given.

Chapter 4

IMPROVING THE FRACTAL CODING / DECODING

As mentioned in the preceding chapter in order to implement the fractal compression algorithm, the set of all possible domain blocks must be searched for each range block in order to find the best match. This process is very time consuming. Since the number of comparisons determines the execution speed, methods for reducing the search time, or even eliminate the search, have been the subject of much recent research [21][31][32]. In this chapter we will investigate some of these improvements to the block oriented fractal compression algorithm in terms of compression time and image quality. In addition some novel schemes will be examined in order to further reduce the compression time without significant loss of image quality.

4.1. Quantizing the Coefficients of the Affine Transform

The quantization process can play a crucial role in the efficient implementation of the fractal compression algorithm. The parameters which describe the affine transformation (i.e. the scaling and the offset factors) are generally quantized and stored compact-

ly, as a constant number of bits. The number of bits directly affects both the compression ratio and the image quality.

In practise, the quantized values of the scale and offset parameters will be used during the range–domain comparison, as well as for the calculation of the RMS error because it has been shown that this gives better results than using exact values and quantizing them later [26].

4.2. Quadtree partitioning

The simple partitioning scheme with a fixed block size illustrated in figures 3.5 and 3.6 is image independent which means that the partitioning time is independent of image complexity. The common geometry and size of range and domain blocks simplifies the whole process of encoding and decoding. Furthermore the storage requirements for the output of the compression is reduced since the size and location of the range blocks can be easily calculated by the decompression program. Nevertheless this scheme has some limitations and does not generally produce optimal compression ratios.

For instance, we can easily observe that real world images are not uniform and therefore they usually possess different complexity from region to region. This means that the distance of the approximation from the original image, might vary radically from one block to another depending on the availability of a matching transformation. In other words some regions of an image might be too complex to be partitioned into 8x8 pixel squares because there might not be a suitable domain block of size 16x16 to closely match the region (for example the eyes of Lena in figure 3.3). On the other hand some larger regions of the image include less detail and they can be covered by larger ranges (i.e. the background of an image) which leads to a reduction of the total number of maps needed.

In order to overcome this limitation an adaptive partitioning scheme is used. The most common of these schemes is called *quadtree partitioning*. Assuming the image size is 256x256 pixels the algorithm initially partitions the image into four equally sized segments (32x32 pixels each). Each of these segments is recursively subdivided into another four equally sized segments, until the quarters are small enough to serve as range blocks (normally of size smaller than 16x16 pixels).

Figure 4.1: *Quadtree partitioning of Lena*

Once the image has been partitioned, the best possible transformation for each block is found and compared with the original block using a distance metric. If the distance between the blocks is lower than a predefined threshold value then the transformation is accepted and stored to the output file. Otherwise the transformation is discarded and the range block is again recurrently divided into four sub–blocks, and the search for each of these quadrants is initiated. This scheme can be continued until either all blocks are covered with an acceptable transformation or until a certain minimum range block size is reached for which the best matching transformation is used.

In figure 4.1 the subdivision of the Lena image is shown using the above partitioning scheme. It is clear that areas with more detail are generally covered by smaller range

blocks than those with less detail. It should be noted here that since the quadtree method repeatedly divides the images in quadrants, it works best with images whose dimensions are equal to a power of 2.

Although with this algorithm some additional parameters for each affine transformation have to be stored, such as the dimension of the range and domain blocks and the quadtree partitioning level, it produces an increased compression ratio and fidelity of the image due to the fewer affine transformations needed to describe the whole image, especially when the image contains large uniform regions.

As far as the compression time is concerned we would normally expect this to be significantly reduced because of the fewer comparisons needed. However the compression time generally increases because the algorithm used in this research initially assumes that a given image region can be encoded with a large range block. Therefore in the case of complex regions these are unsuccessfully compared with the whole set of domain blocks before the region is subdivided to smaller sub–blocks. As each range–domain comparison is computational expensive these unnecessary comparisons produce a significant impact on the fractal image compression time.

More advanced schemes using not only different sizes but also different partition geometries have been implemented. In one of these partition schemes an attempt to select the domain pool in a content dependent way is made (this is called the *HV–partitioning*). Other approaches consider the use of triangles and other polygons instead of rectangles. All partitioning schemes possess several advantages and disadvantages.

For our experiments the choice of the quadtree partitioning is made, because it is simple to implement in a recursive and compact way and has been fully developed and explored. In addition to this partitioning scheme, the use of non–overlapping domain

blocks is adopted because our experiments showed a very small impact in the image quality while the improvement in terms of time is significant.

4.3. Classification Schemes

Regardless of the type of partitioning that is used to generate the domains and ranges, the number of comparisons needed to find a good mapping is the most significant factor that determines the execution speed. The first fractal compression schemes were time consuming because they were trying all the possible comparisons between range and domain blocks and the best matches were used. This is not the case any more. Many efficient methods for reducing the number of comparisons are widely used to implement good encoding algorithms and decreased execution times. Most of these fractal compression algorithms employ a classification scheme. This means that each domain and range block is classified to certain categories according to their characteristics. Each range block is then compared only with domain blocks possessing the same characteristics.

Several types of classification schemes have been proposed [6],[26],[31],[35],[41]. Notably we mention the classification of Jacquin [6] that classifies a sub–image into edge, flat and texture regions. The edge blocks are blocks which contain a distinct edge, the flat blocks contain no distinctive features and textured blocks posses some structure although no specific edge is present. This classification into the three categories considerably reduces the encoding time but the latter still remains very large since the number of the categories is not very large.

A better approach which permits the classification of the blocks into more categories is the one proposed by Fisher [21] [26]. Since this method will be used for the implementation of our experiments it will be described in more detail. The scheme first classifies the blocks into three categories by the distribution of their quadrant's bright-

Figure 4.2: *The three canonical positions for the brightness level of the quadrants of a square sub–image [21]*

ness. More specifically the algorithm takes each sub–image (range or domain) and subdivides it into four quadrants, numbered sequentially as shown in figure 4.2. If the pixel values for each of these quadrants i for $i=1,...,4$ are $r_1^i, \ldots, r_n^i$ where n is the total number of pixels, then the average and the variance values are computed using the following relations respectively:

$$A_i = \sum_{j=1}^{n} r_j^i$$

$$V_i = \sum_{j=1}^{n} \left(r_j^i\right)^2 - A_i$$

The average A_i represents the brightness level whereas the variance V_i the contrast of the sub–image. As can be seen, a square image can always be oriented in such a way so that its quadrants will fall into one of the canonical positions shown in figure 4.2 [21]. These canonical positions define the three major classes which are:

- *Class 1*: $A_1 \geq A_2 \geq A_3 \geq A_4$
- *Class 2*: $A_1 \geq A_2 \geq A_4 \geq A_3$
- *Class 3*: $A_1 \geq A_4 \geq A_2 \geq A_3$

Furthermore, every sub–image within each of the major classes is sub–classified into 24 sub–classes depending on the 24 possible different orderings of the variations in

intensity (contrast) within each quadrant. This results in the creation of a total of 72 classes that can be searched or avoided, depending on speed and quality requirements.

Classifying the sub–images (blocks) by their brightness has the additional advantage of reducing the number of the symmetry operations that have to be tested when we compare one block to another. Indeed, up to seven tests are eliminated as it is no longer necessary to take into account the eight different orientations shown in table 3.2. It is only required to test two classes, one for positive and one for negative scaling because if the scaling value s_i is negative the orderings in the classes shown above are rearranged.

Experiments showed that in practice the consideration of these two different orientations, one for positive and one for negative scaling values, lead to a significant increase in the compression time. Therefore we can simplify the algorithm even more by rejecting the negative s_i and only taking its positive values. This was found to have little effect on the quality of the image. Results for three different test images are shown on the following table:

Image name	**Positive and negative scaling values**			**Only Positive scaling values**		
	Compression time	RMS	PSNR (db)	Compression time	RMS	PSNR (db)
Lena	24.90	9.13	28.91	14.51	10.07	28.06
Collie	23.99	8.88	29.16	14.49	8.70	29.34
S.Fransisco	32.25	11.02	27.28	18.41	12.30	26.33

***Table 4.1**: Compression efficiency for several test images when we consider only positive scaling values or both positive and negative.*

As with any form of classification, it is desirable for a range block to find its best match within the same class so that only one class would have to be searched. Unfortunately results show that when we increase the number of classes searched, we also

increase both the compression time and the quality of the image, because the optimal match is not always in the same class. However, considering the fact that in a real–time distributed video system the loss of the image quality in each frame does not significantly affect the visual outcome whereas the matter of time is very important we will in practice follow the simplified classification scheme described above.

4.4. A Note on Local Self–Similarity

Some of the efforts made to further reduce the number of comparisons and decrease the execution complexity of the compression algorithm concerned the restriction of the distance in the image between the range block and the domain blocks that were tested. This was based on the assumption that a range block is more likely to be similar to a domain block within its neighbourhood due to the existence of some kind of local self–similarity.

However, our experiments have shown this is not the case. Figure 4.3 shows the distribution of the best matched range–domain distance in the image of Lena and the

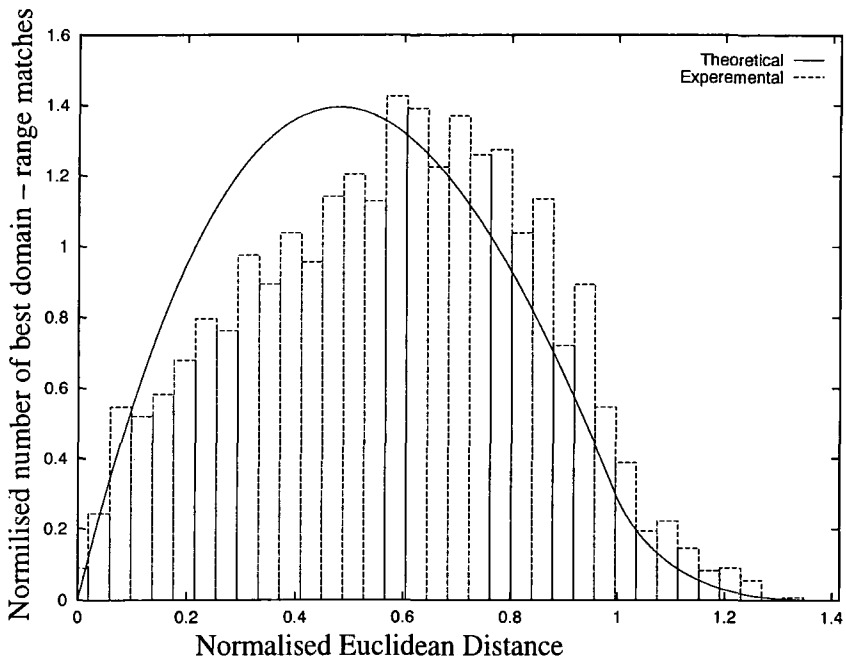


Figure 4.3: Range–Domain distance distribution for Lena image

theoretical distribution of the distance for two randomly chosen points in the unit square as given by Fisher [21]. The figure shows a good match between the two curves, although there is a slight shift of the experimental curve to the right (possibly because the range and domain blocks are not just points). The conclusion that derives from this is that restricting the search to near neighbours will reduce the possibility to find the domain block which best matches to the range block.

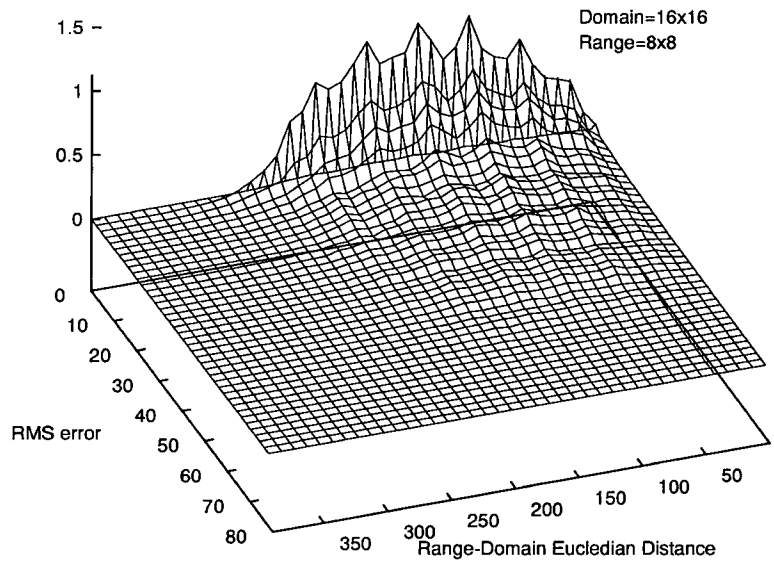


Figure 4.4: *3–D distribution of Lena image, partitioned into ranges of size 8x8.*

Despite the previous conclusion it can not be said that a local search scheme doesn't work at all. Maybe the best possible mapping within the image is not found, but another matching can always be found, good enough to produce an acceptable approximation of the original. In figure 4.4. a 3–D distribution of the range–domain distance can be seen for different RMS error values. It can be observed that although the highest possibility of finding the best matching is somewhere in the middle of the range–domain distance, there is still a great probability to find a reasonable match locally with only a slightly larger RMS error. On the other hand this probability tends to be negligible for the very

large distances. Considering furthermore the benefit of the significant reduction in the number of comparisons, we propose the scheme described in the next section which is based on the partitioning of the whole image into large sections and their autonomous compression as separate images.

4.5. Images as Unions of their Compressed Sections

One of the basic properties of the natural images is that they are closed under the clipping operation [5]. This means that if we choose any rectangular region within such an image, the sub–image that is created is another real world image. Thus it can be compressed separately using the fractal image compression algorithm described above, by partitioning it into range blocks and comparing each of them with only those domains extracted from this sub–image.

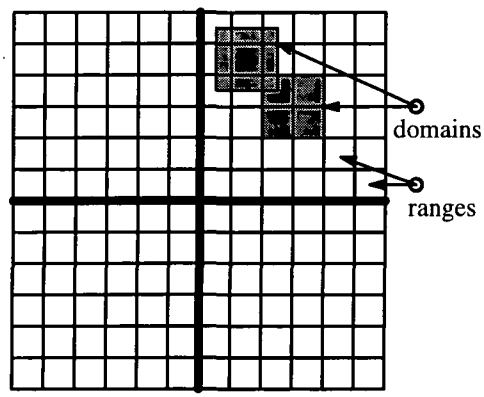


***Figure 4.5**: Partition of the image into sections to reduce the number of comparisons*

In this scheme as a means of speeding up fractal compression, we are using the above property of real world images. The given image is subdivided into several major sections (i.e 4 as shown in figure 4.5) and these are compressed independently as if they were completely unrelated. For each of them we apply a quadtree fractal compression scheme of 3 levels which means that range and domain blocks of three different sizes are used to cover the image depending on the detail provided from region to region. In

addition a classification scheme, such as the one described in the previous section is used.

This segmentation of the image results in a reduction of the number of domain blocks to be compared, since the pool of domain blocks for each part of the image consists of all of the overlapped blocks within the section. By using this algorithm, each of the range blocks are compared with domain blocks in a smaller neighbourhood, thus significantly reducing the number of inefficient comparisons.

The algorithm was tested by partitioning an image repeatedly, producing 4, 16 and 64 major sub–images. The size of the domain blocks varied from 32x32 pixels at the minimum quadtree level up to 8x8 at the maximum quadtree level for each case. The sizes for the range blocks varied respectively from 16x16 down to 4x4. The results of this partitioning are given in tables 4.2 and 4.3 and show significant decrease on compression time. This decrease is not proportional to the number of sections as illustrated on figure 4.6.

Number of sections	Size of section	Compression time	RMS	PSNR
1	256x256	14.51 sec	10.07	28.06
4	128x128	6.54 sec	11.76	26.72
16	64x64	4.49 sec	13.61	25.45
64	32x32	4.43 sec	14.32	25.01

Table 4.2: *The resulting compression time for different partitioning of Lena image*

Number of sections	Size of section	Compression time	RMS	PSNR
1	256x256	14.49 sec	8.70	29.34
4	128x128	6.50 sec	9.68	28.41
16	64x64	4.56 sec	9.98	28.14
64	32x32	4.46 sec	9.60	28.47

Table 4.3: *The resulting compression time for different partitioning of collie image*

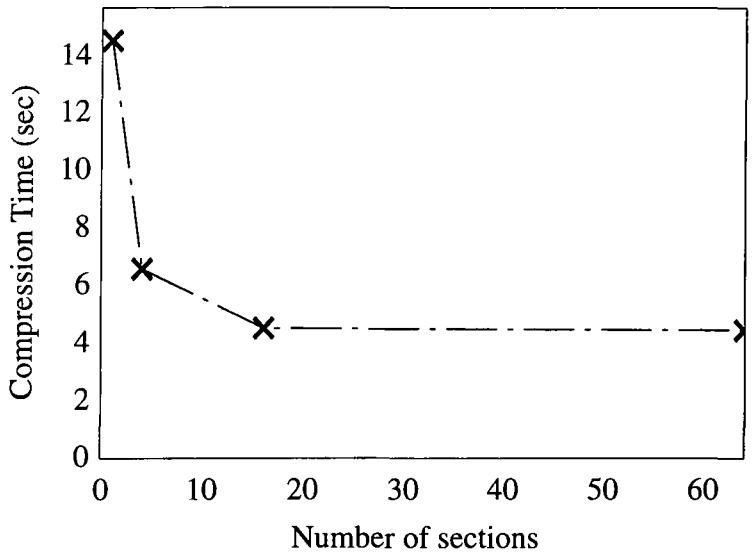


Figure 4.6: *Number of sections versus compression time.*

It seems that after the second partitioning into 16 sections the compression time remains constant. This is because we have already reached a point where only an extremely small number of domain blocks have to be compared with each range block and most of the delay in the execution of the program is caused by the additional comparisons needed for the quadtree partition scheme (as explained above) as well as by the initial decimation and classification of the domain and range blocks. In addition although we have considerably reduced the number of comparisons, each is computationally intensive due to the use of the RMS metric.

Tables 4.2 and 4.3 show that the partitioning also affects the quality of the final image since there are fewer domains available to find a good match. This phenomenon was expected by the analysis of the diagrams in the previous section. However, the rate at which the scheme affects the image quality for three different images, can be seen more clearly in figure 4.7. In this figure the x–axis is a logarithmic scale. It can be seen that the RMS error increases approximately logarithmicaly with the number of sections. This can be explain if we consider the 3–D distribution in figure 4.4. We can see that when we restrict the searching in a very small region around the range block (very

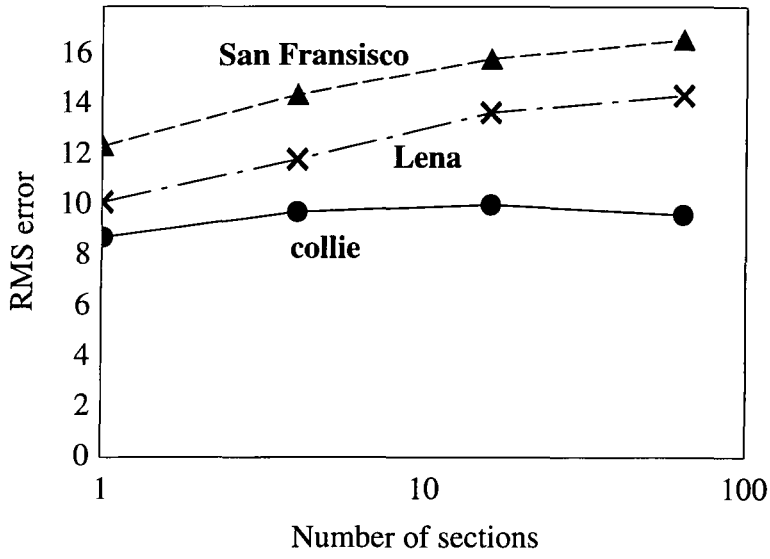


Figure 4.7: *Number of sections versus RMS error*

small euclidean distance), the probability to find a good range–domain match is decreasing significantly.

The images that result when this scheme is applied to the original Lena image (figure 4.8) using a partitioning of the whole image with a different number of sections, is shown in figures 4.9, 4.10, 4.11. Visually there is not a significant loss in detail when comparing the uniformly compressed image of figure 4.9, to the image of figure 4.10 which is the union of the four independently compressed sub–sections. The gain in terms of compression time however is very large since the time needed for the compression is reduced by more than 50%. Similar conclusions can be drawn from a comparison of figures 4.9 and 4.11, although the loss in image quality is now more obvious and the decrease in compression time is not so great.

Another benefit of the scheme is that larger compression ratios can be achieved since a smaller number of bits is needed for the storing of the information related to the location of the domain block. More over and most important for this work, this method can be readily parallelized as we can compress each section on separate processors sharing the same memory.

A similar scheme has been proposed in the literature by Dudbridge in [24]. His aim was to greatly reduce the coding time of an image. In order to achieve this he only uses blocks that are directly adjacent, but complexity is added by using a higher order

Figure 4.8: *Original image*

Figure 4.9: *Compressed as one image*

Figure 4.10: *Compressed as 4 independent sections*

Figure 4.11: *Compressed as 16 independent sections*

intensity scaling function. His results are encodings in less then one second but there is a great cost for the compression ratio and also the image quality is poor.

4.6. Resolution and Hierarchy

As mentioned before the Iterated Function System (IFS) describes the relations between different sub–blocks of the image, namely domain blocks and range blocks. In order to efficiently describe this relation the IFS does not need to contain any particular information on the size of these blocks. Instead the ratio of the domain to range size, *D/R*, and the total number of range blocks, N_R, fitted in the image must be known. The ratio is defined in most of the cases to be 2 which means that the domain block size is taken to be double the size of the range in order to ensure contraction. Therefore, decoding an IFS code can result on different sizes of the final image depending on the size of the range block that will be considered.

For example, let the size of each side of the range blocks to be $R = N / M_R$ where N is a physical number and M_R is the number of range blocks on which the image was partitioned. In this case the resulting image will be of size *NxN*. On the other hand if the range size is chosen to be $R = N / 2M_R$ the resulting image will be half the size of the previous one. If the size of the range block is chosen to be larger than the size of the range blocks on which the original image was tiled in order to produce the IFS code then a the fractal algorithm creates artificial information. This advantage of the fractal algorithm which occurs when a magnified copy of the image is produced is demonstrated in figure 4.12. The pixelisation phenomenon is easily observed when the image is normally magnified. On the other hand the fractal scheme smooths the previous phenomenon producing a magnification of higher quality.

The previous results might seem to be contradictory to the contraction mapping theorem which states that in a complete metric space, for every IFS there is only one

(a) *Normal Zoom* (b) *Fractal Zoom*

Figure 4.12: *A magnified region of the baboo image.*

and unique corresponding attractor. However the different attractors (in size) shown in figure 4.13 (a),(b) and (c) can be considered to be attractors on different spaces, for instance $\mathbb{R}^{64}$, $\mathbb{R}^{128}$ and $\mathbb{R}^{256}$ respectively. Assuming for simplicity that the different image attractors are one dimensional fixed point vectors, we can define in more mathematical terms the relation amongst them as follows[23]:

Definition: Assume $f \in L^{\infty}[0,1]$. Than we can define a function $f_r(i)$ such that:

$$f_r(i) = r \int_{(i-1)\frac{1}{r}}^{i\frac{1}{r}} f(x)dx$$

where $f_r(i)$ is said to be the function $f(x)$ at resolution r and $i=1,...,r$. *The* function $f_{r1}(i)$ is said to have a higher resolution than the function $f_{r2}(i)$ when $r_1 > r_2$.

Theorem: For every IFS code, there is a unique function *f(x)* such that an image vector serves as the attractor of the IFS if it is equal to the function *f(x)* at resolution $r = N \equiv M_R \ /2$:

$$V_N(j) = f_N(j), \qquad j = 1,\ldots,N$$

The proof of the this theorem [25] produces the following corollary:

$R = \frac{N}{M_R} = \frac{64}{32} = 2$

(a)

$R = \frac{N}{M_R} = \frac{128}{32} = 4$

(b)

$R = \frac{N}{M_R} = \frac{256}{32} = 8$

(c)

Figure 4.13. *Decoding of the same IFS on different spaces*

Corollary: Let V_N be a fixed–point of a given IFS code in the $\mathbb{R}^N$ space. Then the $V_{\frac{N}{2}}$ will be a fixed point of the same IFS but in the $\mathbb{R}^{N/2}$ space if the following equation is in effect [23]:

$$V_{\frac{N}{2}}(l) = \frac{1}{2}(V_N(2l-1) + V_N(2l)), \qquad l = 1,\ldots,\frac{N}{2} \tag{4.1}$$

Proof:

$$V_{\frac{N}{2}}(l) = f_{\frac{N}{2}}(l) = \frac{N}{2}\int_{(l-1)(\frac{2}{N})}^{l(\frac{2}{N})} f(x)dx =$$

$$= \frac{N}{2}\int_{((2l-1)-1)(\frac{1}{N})}^{2l(\frac{1}{N})} f(x)dx =$$

$$= \frac{1}{2}\left[N\int_{((2l-1)-1)(\frac{1}{N})}^{(2l-1)(\frac{1}{N})} f(x)dx + N\int_{(2l-1)(\frac{1}{N})}^{2l(\frac{1}{N})} f(x)dx\right] =$$

$$= \frac{1}{2}\left(f_N(2l-1) + f_N(2l)\right) =$$

$$= \frac{1}{2}\left(V_N(2l-1) + V_N(2l)\right)$$

The final equation denotes the relation of a coarser resolution image vector to its finer resolution one (zoom out) and in practice states that in order to compute an element of $V_{N/2}$ from a given vector V_N the average of two adjacent elements of V_N have to be taken. On the other hand when we want to calculate the finer resolution vector V_N by a coarser one (zoom in) the following equation has to be applied:

$$V_N((k-1)R_N + n) = s_k V_{\frac{N}{2}}((m_k-1)\ \delta D_{\frac{N}{2}} + n) + o_k \tag{4.2}$$

where $k = 1,\ldots,M_R$ and $n = 1,\ldots,R_N$

The R_N and M_R parameters define the size and the number of the range blocks respectively whereas the $\delta D_{\frac{N}{2}} \equiv \frac{\delta D_N}{2} = \frac{R_N}{2}$ defines the shift between consecutive domain blocks and m_k indicates the domain block which best mapped the *k–th* range block. The parameters s_k and o_k are the scaling and offset factors. The above equation is proved in [25]. We can easily observe that the above equation is very similar to the transformation W_N which is used to create the fixed–point V_N iteratively by domain blocks taken from an initially arbitrary image of the same resolution as described in chapter 3. The difference is that with transformation (4.2) each iteration produces vectors of a higher resolution

Both equations (4.1) and (4.2) can be used to describe the relation not only between the aforementioned vectors but also between the vectors $V_{N/2}$ and $V_{N/4}$ and so on. Thus a *hierarchical* structure is created which derives naturally due to the self–similarity property of the fractal structures. An example of this hierarchy is shown in figure 4.13 where the decoded images (a), (b) and (c) are derived for different sizes of range blocks. In this hierarchical structure the higher level is considered to be the image with the coarsest resolution (figure 4.13 (a)). In order to find the smallest possible attractor size which can serve as the top level, all we have to take into account is that the smallest range block size is at least 2x2 pixels. Therefore the baboon image in figure 4.13 (a) is the smallest possible one.

4.7. Speeding Up the Decompression

Although the process of decompression for a fractal coder is relatively fast there are several techniques that can be applied in order to speed it up even more and therefore approach real–time execution on a typical Personal Computer or workstation. One of the most efficient algorithms is attributed to Z.Baharav et al. [23][25] and is based on

the previously described fractal characteristic of resolution independence and the hierarchical structure which derives from this.

In particular the main idea is to apply the iterations needed for the decompression in such a way that a reduced version, of the original image is produced which can be resized to its original size simply by using the two dimensional equivalent of equation 4.2. Indeed as shown in figure 4.14 the time needed for the reconstruction of the image is almost proportional to the size of the reconstructed image. Thus if we could apply the computationally highly demanding iterations needed for the decompression to a reduced version of the image and then rescale it to its original size, this could considerably reduce the number of calculations needed and the execution of the algorithm could be made much faster.

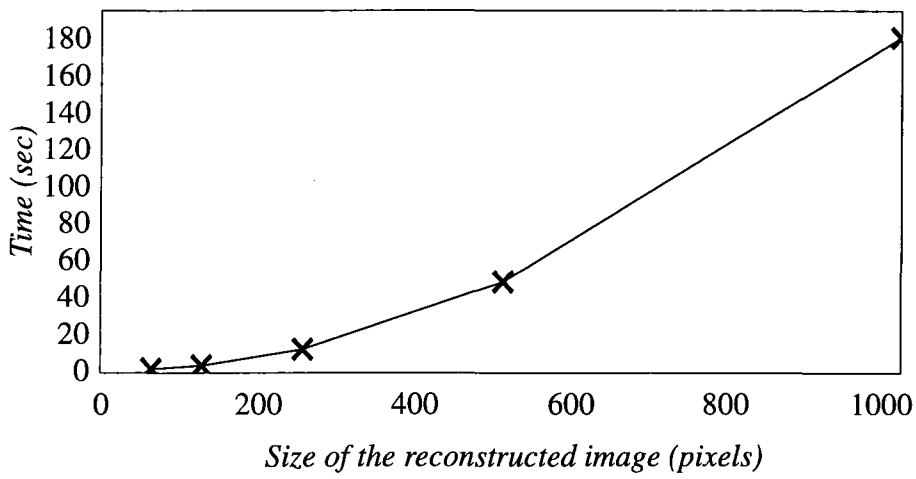


Figure 4.14. *Decoding time versus size of the reconstructed image using the conventional decompression algorithm.*

The resulting improvement in the speed of execution of the decompression algorithm is shown in figure 4.15. The execution time is reduced significantly and becomes less than one second for image sizes up to 256x256. This gain becomes greater as the size of the reconstructed image increases. The rescaling from the top level attractor in the

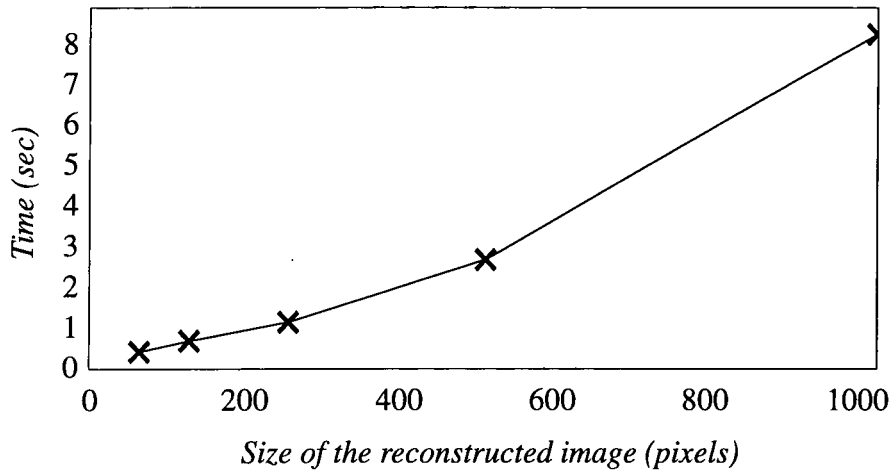


Figure 4.15. *Decoding time versus size of the reconstructed image using the hierarchy based decompression algorithm.*

hierarchy to the desirable final size achieved in such a way occurs without any further loss of detail as shown in figure 4.16.

4.8. Choosing a different metric

As we mentioned before the choice of the metric is of great importance for fractal compression. Depending on the metric, a more rapid contraction, or no contraction at

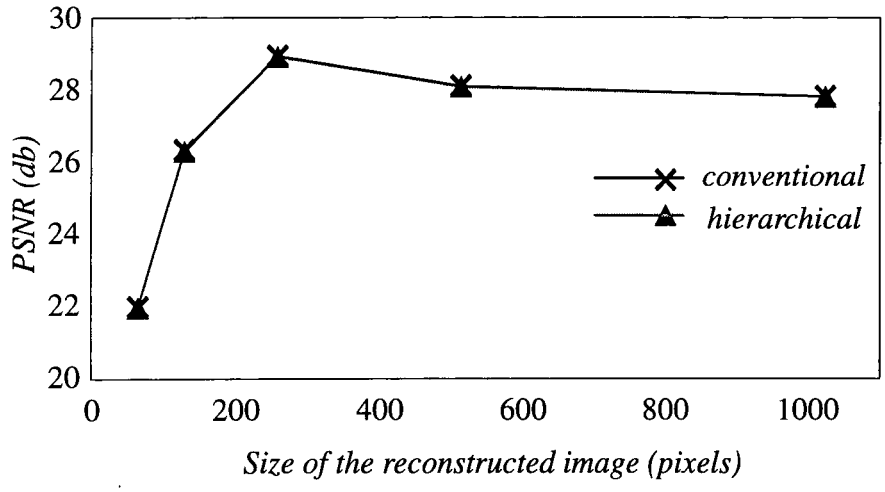


Figure 4.16. *Comparison of the PSNR versus size of the reconstructed image diagram for both hierarchical and conventional decompression algorithms.*

all, of the transformations can be determined. Until now the RMS error has been used because of the several advantages it provides. A significant advantage is that the scale s and offset o parameters can be easily calculated as part of the range–domain comparison process using the equations (3.10) and (3.11) of chapter 3.

However the calculation of the RMS error contributes significantly to the execution time of the algorithm since the routine that compares the range–domain blocks has been found to be by far the most time consuming process. Therefore a different metric is proposed in order to overcome the computational complexity of the RMS error. This metric was the *median metric* which is inherited from the L^1 norms and is defined as follows:

$$d(f,g) = \sum_{x=1}^{N}\sum_{y=1}^{N} | f(x,y)-g(x,y) | \qquad (4.3)$$

where N is the total number of pixels vertically and horizontally and f and g denote functions representing the sub–images to be compared.

This metric derives from the generic form of the L^p metrics [22]:

$$d_p(f,g) = \| f-g \|_p = \left[\int_{I^2} | f-g |^p \right]^{\frac{1}{p}}$$

when $p=1$, whereas the L^2 metric (for $p=2$) corresponds to the RMS error.

In order to calculate the complexity of the new metric we will define S *as* the computation time of one summation/subtraction and M the time of one multiplication. Than the total computation time needed for the calculation of the RMS error will be:

$$t_{RMS} = N^2(S + M)$$

whereas for the median metric the same quantity will be only:

$$t_{med} = N^2 S$$

However when we use the metric of equation (4.3) to make the comparisons between the range and domain blocks the problem of calculating the offset and scaling parameters arises. It is obvious that since the equation is of first order there is no minimum and therefore we can not minimize the partial derivatives with respect to s and o as we did with the RMS metric in chapter 3.

In order to solve this problem, the use of a predefined constant value for the contrast setting was proposed. The contrast is represented by the scaling parameter such that $0<s<1$ in order to ensure contraction. In practice a value of $s=0.25$ has been found to be a reasonable value to use, as for larger scaling values the reconstructed image is darker than the normal one and vice versa for smaller scaling values. The offset parameter represents the brightness setting and thus the difference in the brightness of the two blocks that are compared can be considered to be the difference of their average pixel intensities, as shown in equation 4.4:

$$o = \left| \frac{\sum_{x=1}^{N}\sum_{y=1}^{N} f(x,y)}{N^2} - \frac{\sum_{x=1}^{N}\sum_{y=1}^{N} g(x,y)}{N^2} \right| \qquad (4.4)$$

where the f(x,y) and g(x,y) represent the range block and the decimated domain block which has the same size with the range respectively. Again the o parameter is quantized in order to be stored more compactly.

Unfortunately, in practice the process did not appear to perform well. The major problem was that the reconstructed image contained pixelisation artifacts as can be seen on the Lena image of figure 4.17(a). This phenomenon is a natural consequence of the choice of constant s parameter. In order to eliminate this, very small range blocks should be considered to tile the image (2x2, which is the smallest possible). In this case

(a) Minimum Range size 4x4 *(b) Minimum Range size 2x2*

Figure 4.17. *Lena encoded using the median metric.*

the quality of the reconstructed image becomes extremely high as can be seen in figure 4.17 (b), and the pixelisation phenomenon becomes negligible. However, with such small range blocks the compression ratio is very low and the compression time becomes large.

A further problem is that in order to use the median metric we have to calculate the absolute value of the image difference and this process appears to be almost as computationally demanding as the multiplication process. Thus we don't finally expect any considerable improvement in terms of compression time.

Comparative results between the RMS metric and the median metric can be seen in table 4.4. for different image tiling. The choice of the tolerance for both metrics did not seem to influence significantly the quality of the final image, thus a relatively large tolerance value was chosen for both cases, in order to improve the compression time and compression ratio. The table shows that the fractal compression algorithm based on the RMS metric can achieve higher compression ratios and better quality even at coarser tiling of the image.

Size of Ranges		RMS metric			Median metric		
Min	Max	Comp. Time	Comp. Ratio	PSNR	Comp. Time	Comp. Ratio	PSNR
32x32	8x8	7.53 s	26.71	24.19	5.93 s	28.89	20.91
8x8	8x8	9.44 s	22.33	25.28	6.11 s	27.38	21.31
16x16	4x4	26.08 s	9.10	28.91	19.63 s	6.36	23.97
4x4	4x4	36.59 s	5.12	29.57	20.88 s	6.39	23.98
8x8	2x2	219.49 s	3.97	34.49	234.27s	1.68	27.17
2x2	2x2	889.36 s	1.18	42.13	257.52 s	1.50	28.06

***Table 4.4**: RMS–Median metric comparison for different partitionings.*

The superiority of the RMS metric becomes apparent in figures 4.18 and 4.19. Figure 4.18 shows the compression time versus PSNR for the two metrics. The vertical axis is in logarithmic scale in order to illustrate more efficiently the difference of the two curves. It can be seen that better signal to noise ratios can be obtained for the same compression time when the RMS metric is used. On the other hand, figure 4.19 shows the compression ratio versus PSNR for both metrics. Again, larger compression ratios for the same signal to noise ratio are achieved with the RMS metric. Therefore the RMS metric appears to be the better choice of metric for fractal algorithm.

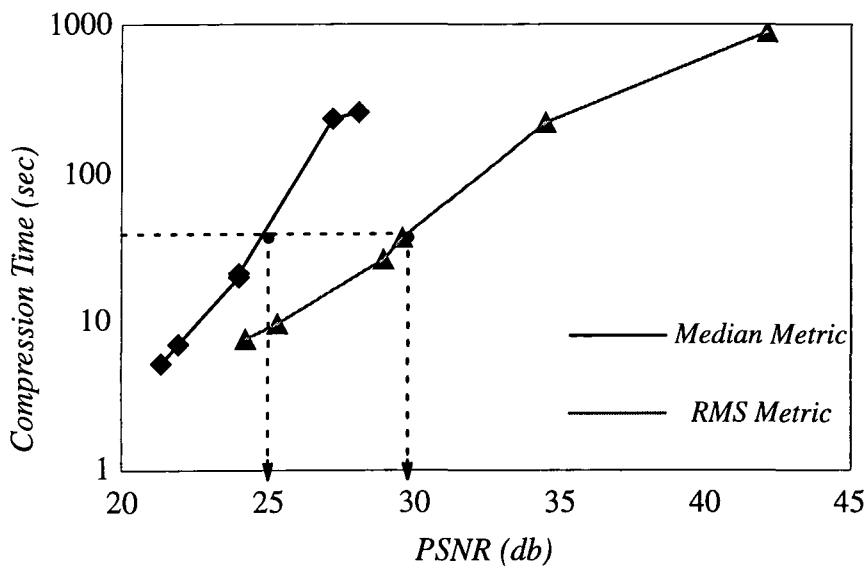


***Figure 4.18**: PSNR versus compression time for both median and RMS metrics.*

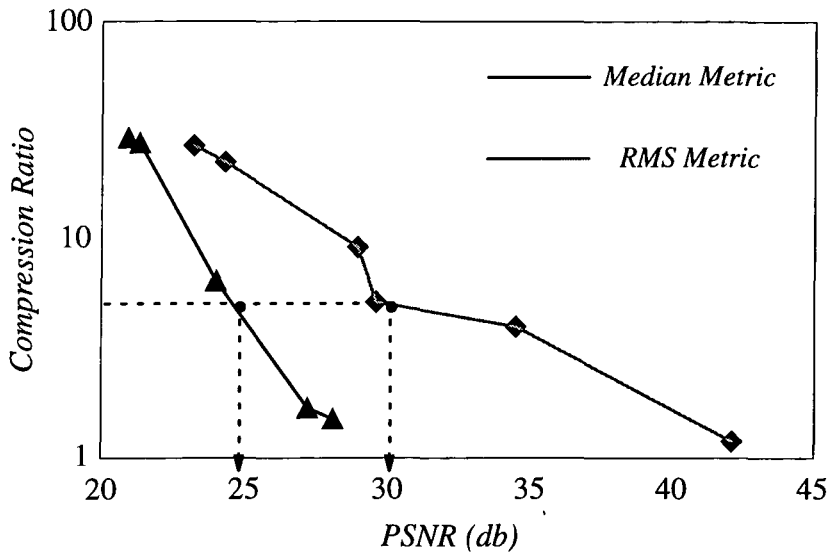


Figure 4.19: *PSNR versus compression ratio for both median and RMS metrics.*

4.9. Summary

In this chapter, several improvements of the fractal compression and decompression algorithms were introduced. The first improvement was the quantizing of the coefficients of the affine transform, in order to obtain more compact storing, using less bits per coefficient. In addition, several adaptive partitioning schemes were discussed in order to improve the image quality and the compression ratio. Amongst them the quadtree partitioning was chosen for our implementation since it is relatively simple to implement in a recursive way and performs well. At this stage, the use of non–overlapping domain blocks was adopted, since this was found to have a small impact in the image quality while the decrease of the encoding time was significant.

In order to reduce the number of range–domain comparisons, which greatly influences the execution time, a classification scheme was considered. This scheme was classifying the range and domain blocks into 72 classes according to the average pixel intensity and the variance of their quadrants. Furthermore, a scheme where the image was considered the union of its compressed sections was proposed. In this scheme the

image was partitioned into a certain number of sections (4, 16 or 64) and each section was compressed individually as if they were individual images. The results showed significant decrease in the execution time but this also influenced the image quality. The image quality was reduced significantly specially when a large number of sections (64) was considered, since (as it was proved) images do not imply local self–similarity.

The hierarchical interpretation of the image resolution was represented in the next stage. This property was used to speed up the decompression process. The time consuming IFS iterations on an arbitrary image were applied to produce a reduced version of the image and then this version was rescaled without any loss in detail at its normal size. The latter resulted in significant decrease of the execution time.

Finally the use of the median metric as an alternative to the RMS was investigated. Results showed that the RMS metric significantly outperformed the median metric, therefore the RMS was finally adopted for the implementation of the fractal compression algorithm.

Despite the significant improvement obtained using the above techniques in image quality, compression ratio and execution time (for both compression and decompression), the compression time remains relatively long. Therefore, in the current implementation it can not be used to encode video frames in real–time and other techniques have to be considered to further reduce it. Such techniques are investigated in the next chapter.

Chapter 5

FRACTAL CODING OF IMAGE SEQUENCES

Having examined several techniques to improve the fractal image compression algorithm we will now attempt to apply and optimise these techniques for image sequences, namely video. As mentioned in the previous sections the fractal coding of still images has proven to be promising and efficient. Even more, its properties seem to satisfy many of the requirements for video streams. It provides for example fast and hierarchical decompression by progressively improving the images as part of the basic algorithm and it is scalable to arbitrary image sizes with good results.

Considering these advantages and the continuously increased demand for video coding we can explain the efforts that have been developed recently to extend the algorithm for compressing image sequences. It is the aim of this chapter to investigate several of these efforts in order to implement the video encoding as applied in distributed systems more efficiently, whereas the decoding process will approach as much as possible the real time execution.

5.1. Volume Fractal Coding of Video Frames

Three major types of fractal video coding have been reported in the literature. The simplest one is to encode every frame individually. This means that similar to the

Motion–JPEG compression method, every frame is considered to be an intra–frame. However, this technique is inefficient because the correlation between the consecutive frames is not taken into consideration. Therefore the temporal redundancy remains and the very low bit–rates demanded by the video applications cannot usually be met. Furthermore the fast compression of each frame needed for real–time applications is almost impossible to achieve by this approach.

Another method for fractal video compression involves the extension of the algorithm into three dimensions where the third dimension is considered to be time. The video sequence is then compressed and decompressed as one large volumetric data. Not many approaches of this type have been reported in the literature but one method is reported by Lazar et al [27]. This true three dimensional compression implies a set of 3D domain cubes being mapped onto 3D range cubes. The whole process is based on a three dimensional transform which can be specified using only a few more bits than its two dimensional equivalent. Furthermore it allows the encoding of a large number of 2D range blocks using the same transformation. Essentially the algorithm is using prediction to find the best mapping, not necessarily for the current frame but instead for a series of frames.

In particular the algorithm of Lazar [27] operates on what he has termed R–frames and takes domain blocks from the D–frames. The range cubes are portions of the R–frame area while the domain cubes do exist within the D–frame area. The R–frame and D–frame areas have to end on the same frame but the D–frame area may begin before of the R–frame area as shown in figure 5.1.

An adaptive partitioning scheme appropriate for 3D video blocks is also represented by Lazar et al.[27]. This scheme is summarized in figure 5.2. In essence when the algorithm can not find a good mapping, instead of splitting the range cube into eight identical smaller cubes it splits it either spatially into 4 smaller blocks or temporally

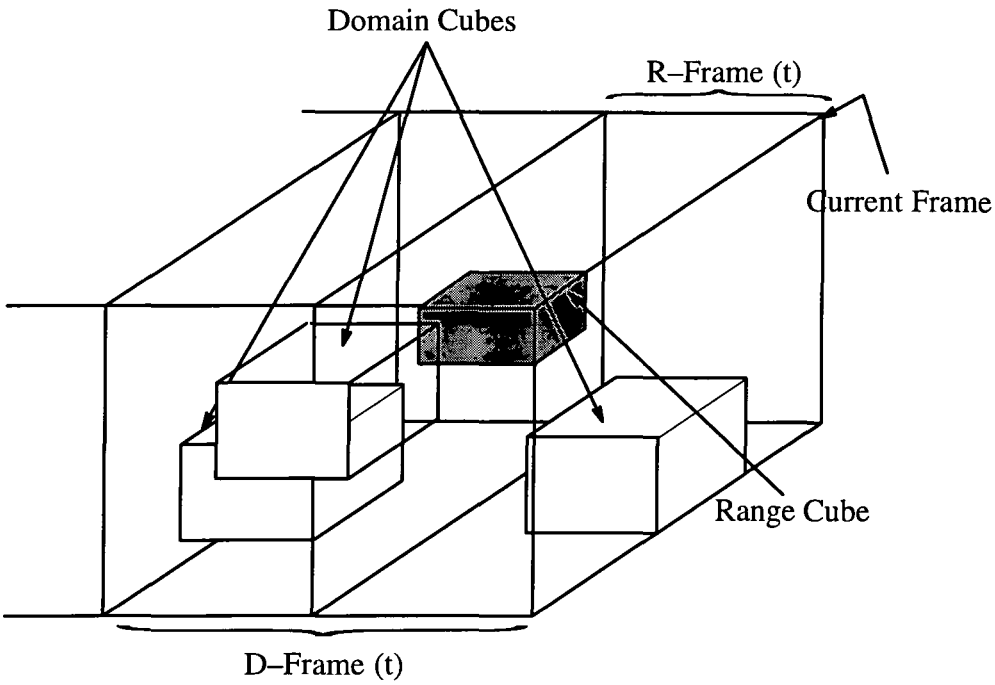


Figure 5.1: *Tiling of the Video Signal into Domain and Range Cubes [27]*

into 2 blocks. The choice of partitioning is determined by the error, if it is evenly or unevenly distributed along the block respectively.

These fractal video schemes which treat the video signal as a volume data and the time domain as a third spatial dimension have several disadvantages. First of all such

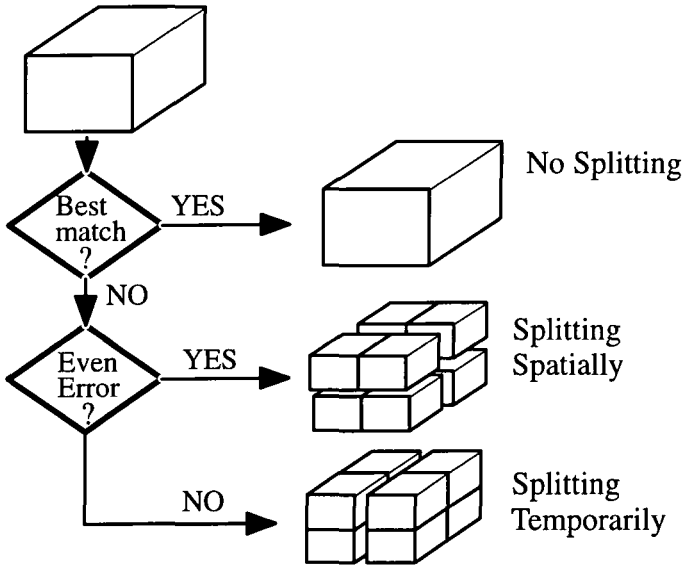


Figure 5.2: *3–D adaptive partitioning [27]*

schemes demand huge amounts of memory to serve as a frame store since during the decoding several frames have to be stored in order to create the necessary volume data. Furthermore the results of the early investigations are not encouraging with regards to image quality and computational requirements.

For instance, the additional third dimension increases significantly the number of the possible orientations and as a consequence the complexity of the algorithms becomes very large. Some efforts have been made to reduce these orientations by considering that in most of the natural video sequences many of these orientations do not make sense. Therefore it can be beneficial to restrict to separate spatial and temporal transformations. However the problem of the high complexity still remains in comparison to the 2D scheme. The decompression process might require as well multiple iterations to achieve convergence.

A real–time distributed video through a telecommunication network system requires for the signal to be continuously compressed and decompressed and not transmitted as a whole. Therefore other schemes based on the still fractal image coding will be examined in more detail.

5.2. Still Fractal Image Coding for Image Sequences

The most commonly used type of video coding involves the application of fractal techniques such as the ones already described in the previous chapters for the coding of still images, to the video stream. Unlike the simple fractal coding described above, which considers each frame individually, the approaches of this type take into account the correlation between the consecutive frames and apply several techniques in order to reduce the temporal redundancy.

For instance, in [28] Reusens suggests a tri–dimensional fractal video compression technique. This approach is based on an appropriately extended affine transformation

defined as the composition of a geometric (spatial) part and a massic (temporal) part. The temporal part of the transformation has the following form:

$$T_i\begin{bmatrix} x \\ y \\ t \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 & 0 \\ c_i & d_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s_i \end{bmatrix}\begin{bmatrix} x \\ y \\ t \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ o_i \end{bmatrix}$$

where the s and o parameters represent again the contrast and the brightness shift respectively. A three–dimensional adaptive partitioning of the sequence, similar to the one used in [27], is adopted in order to enable one to vary in an optimal way the size of the blocks according to their temporal and spatial statistical properties.

Another, simpler attempt, to reduce the temporal redundancy is called conditional block replenishment. This scheme uses constant blocks for those areas that can be sufficiently covered by them and transmits mappings only for those blocks that are sufficiently complex.

Such a simple scheme was investigated in order to implement our video system. Each original frame was tiled adaptively using a common partitioning scheme such as the quadtree partitioning. Before we start searching for the best range–domain mapping, every range block is compared to the corresponding range block of the previous original frame. The comparison is made by calculating the RMS distance of the two blocks. If this distance is found to be small enough (less than a predefined tolerance value) then a specific character is stored which will inform the decompressor that this range block can just be copied from the previously reconstructed frame. Otherwise the algorithm continues and searches for best range–domain mappings where the pool of domains is created by domains of the previous original frame. This can be considered as a crude form of motion detection.

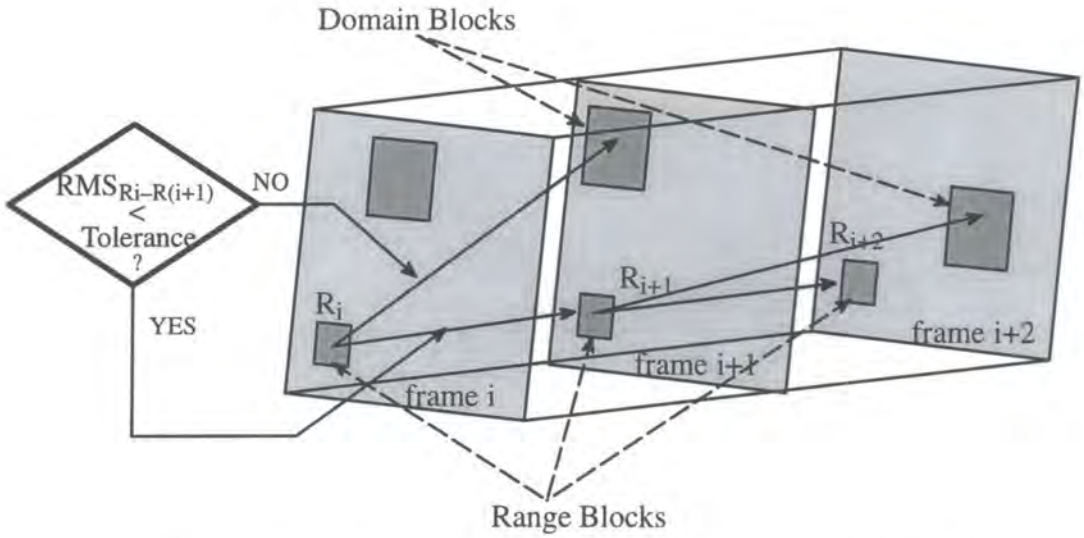


Figure 5.3: *Comparison of corresponding range blocks to reduce temporal redundancy*

The scheme is represented graphically in figure 5.3. If the algorithm can not find a good range–domain mapping, the range block is subdivided into four quadrants and for each one of them the previous process is repeated (quadtree partitioning). Although the basic scheme has been already described there are still several points that require some more discussion. One of them regards the image reconstruction process.

Generally, as mentioned in the previous chapters, a standard number of iterations has to be used for the reconstruction. This must be large enough to ensure convergence since the cost of testing for convergence would outweigh the savings of having fewer iterations. Therefore the process suffers from relatively long decompression times unless we use hierarchical representation or other complex algorithms which speed it up. In order to overcome the problem in a more natural way we will consider the use of the previously decompressed frame as the initial image. When this is done, each frame can be decoded successfully, and in most cases after just one iteration since the consecutive frames have a high correlation. This speeds up the decompression process even more and makes unnecessary the hierarchical reconstruction described in the previous chapter.

Another point that has to be discussed regards the first frame in the sequence. This image can not be encoded as described above, or reconstructed in the normal way because its predecessor in both cases is an empty, uniformly black image. In practice there are two possible options that can be applied:

- The first frame can be compressed as a still image, using a conventional still image compression technique.
- Alternatively we can just consider as initial input image in the decompressor, the first frame of the image sequence which will be distributed together with the IFS code of the first frame.

Although the latter method has the advantage of starting the process from an original reference frame, which will positively influence the quality of the next frames, it decreases the total compression ratio of the video signal. Furthermore, the resolution independence property of the algorithm is lost since the reconstruction can only be made at the size of the original image.

In practice we will implement both of these schemes in order to extract useful conclusions regarding the compression and decompression time, and the quality of the reconstructed frames.

5.3. Implementation and Results

Experiments were performed on two different image sequences, namely the *"Trevor"* and *"Miss America"* sequences, both of them being talking heads. The choice of these sequences was made because of the relatively small motion they provide, the large uniform background areas they include and also because these kind of video frames dominate on a video–conferencing system.

The video sequences are considered in practice as sequences of functions $f_1, f_2, \ldots, f_n$ where n is the total number of frames. These functions have the form described in

section 2.1. At each stage, the current original frame f_i is tiled into range blocks using a quadtree partitioning scheme of two levels. Consequently the block size may vary from 16x16 to 4x4. The range blocks are then characterized as *static* and *dynamic*:

- The *static* range blocks are those blocks which remained almost identical compared with the corresponding area in the previous frame f_{i-1}; therefore we just copy this area directly from the predecessor.

- The *dynamic* range blocks are those blocks which have changed significantly compared with the corresponding blocks of the predecessor f_{i-1}, due to the motion of an object. For these ranges, a good range–domain mapping, for which the rms error is below a certain *encoding threshold* t_f, has to be found. The pool of domains is constructed from domain blocks in the previous frame f_{i-1}, which are twice the range size. In order to reduce the comparisons both the ranges in frame f_i and domains in f_{i-1} are classified in the 72 classes as described in the previous chapter.

The characterization of the range blocks is based on the calculation of the Mean Square error between the corresponding ranges in consecutive frames:

$$\Delta R_i = \frac{1}{M_X M_Y} \sum_{x=1}^{M_X} \sum_{y=1}^{M_Y} \left(R_i^{f_k}(x,y) - R_i^{f_{k-1}}(x,y) \right)^2 \qquad (5.1)$$

where M_X and M_Y are the numbers of pixels of the range block horizontally and vertically, and f_k and f_{k-1} represents the *kth* and *(k–1)th* frame respectively. If the ΔR_i error is below a certain threshold, ε_f, then the range block is static otherwise it is taken as dynamic. We will refer to the ε_f threshold as the *temporal tolerance* and its choice will be set empirically throughout our experiments. On the other hand, the encoding threshold t_f will be set in a constant value $t_f=64$ for most of our experiments in order to evaluate the contribution of the temporal tolerance on the results.

The scheme which compresses and decompresses the first frame taking into account the first original one was initially considered. In figure 5.4 and 5.5 we can see the fluctuation of the PSNR by frame for both the Trevor and Miss America sequences respectively. The three curves in the same diagram represent the fluctuation for different compression ratios which resulted for various temporal tolerance settings. The PSNR seems to be highly correlated to the value of the tolerance.

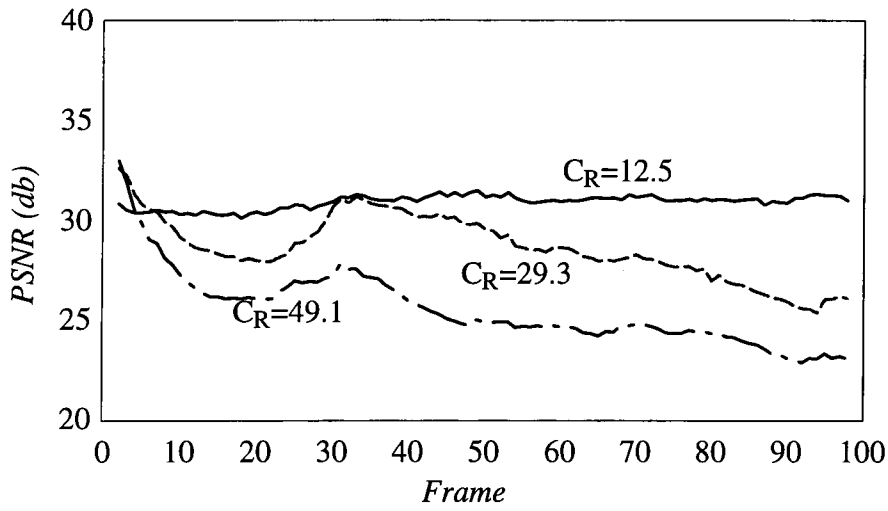


Figure 5.4: *PSNR fluctuation by frame of the Trevor sequence for different compression ratios. As first frame is taken the first original one.*

Large tolerance settings, result on smaller average PSNR and vice versa. In addition it can be seen that for large values of the tolerance setting, the first few frames have high image quality due to the original first frame which was used as reference point. However the image quality is significantly reduced until around the 10th frame. After that point the image quality continues to decrease, although sudden increases may be observed as well, depending on the motion of the talking head. Particularly in the case of many sudden movements this problem becomes even more intensive and the method suffers by the drawback of unpleasant visual artifacts. When a block that was being encoded with a constant block suddenly needs a mapped block, then there is an obvious

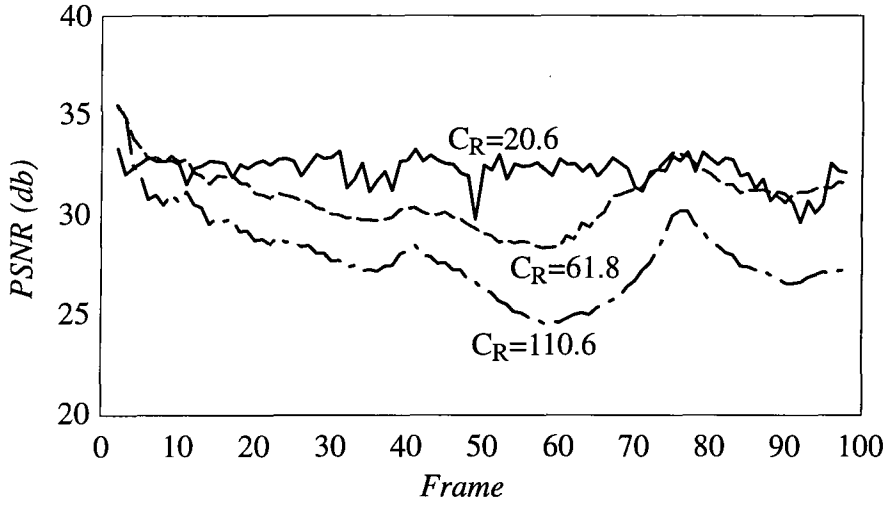


Figure 5.5: *PSNR fluctuation by frame of Miss America sequence for different compression ratios. As first frame is taken the first original one.*

change in the visual appearance of the decoded block. Such intensive phenomena result in unacceptable video signals.

On the other hand when we set very small temporal tolerance values the areas of the frame that are covered by static ranges are very small and each frame is in fact decompressed as a still image. This results on an almost constant quality throughout the decompressed signal and the considered original first frame does not seem to have any effect on the signal quality. The visual artifacts described above are not intensive anymore and they tend to be completely eliminated as we approach tolerance values close to zero.

The disadvantage of setting a very small temporal tolerance value however is that this results in relatively larger compression and decompression times as can be seen in figures 5.6 and 5.7 respectively. These figures show results for the two different video sequences mentioned above. In figure 5.8 the variation of the compression ratio with respect to the temporal tolerance value is shown whereas in figure 5.9 the average PSNR of the 100 first frames of the sequences versus the temporal tolerance can be

seen. In every case the value of the temporal tolerance seems to influence significantly these parameters.

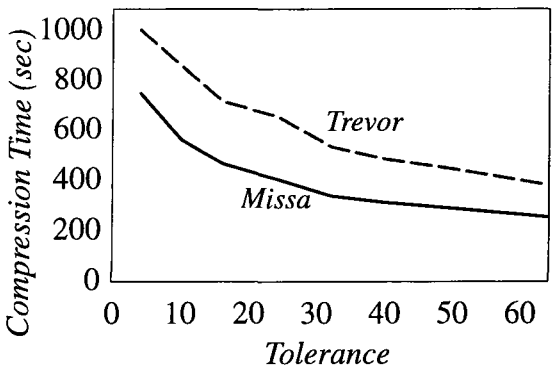


Figure 5.6: *Compression Time versus Temporal Tolerance.*

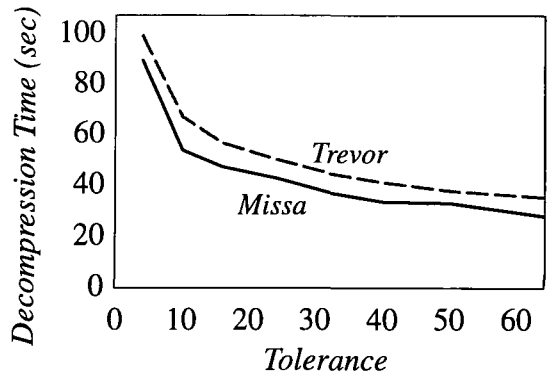


Figure 5.7: *Decompression Time versus Temporal Tolerance.*

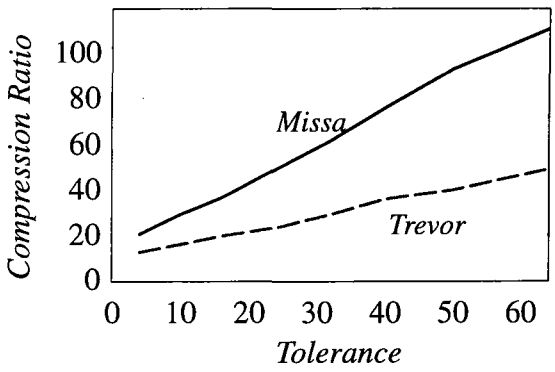


Figure 5.8: *Compression Ratio versus Temporal Tolerance.*

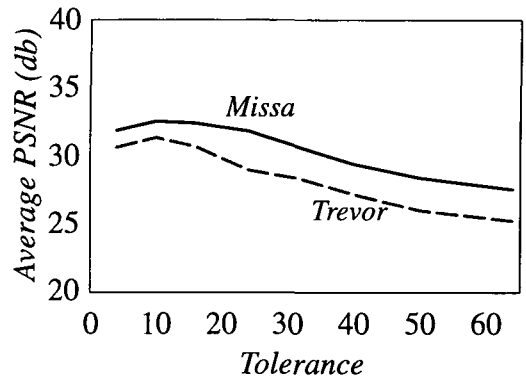
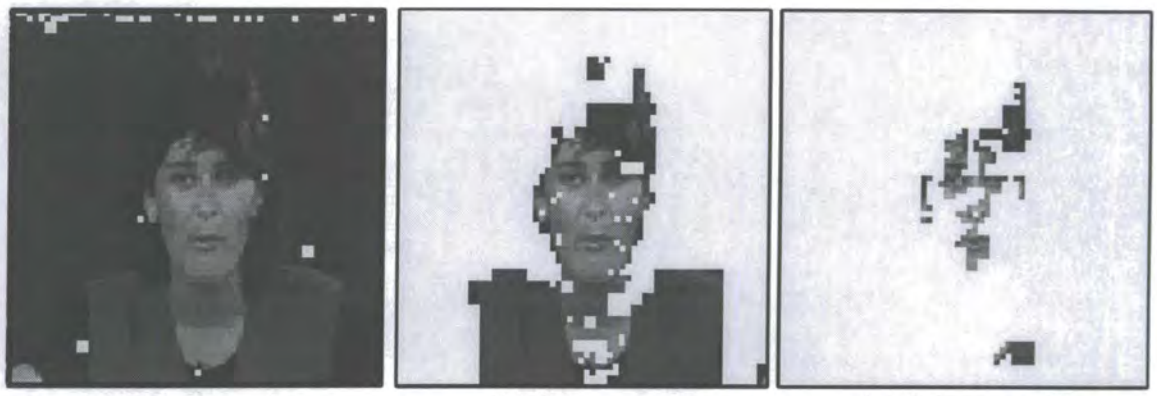


Figure 5.9: *Average PSNR versus Temporal Tolerance.*

The latter can be explained if we compare the total area (white area) of the frames that is covered by static ranges, for different tolerance settings. This is shown in figure 5.10. For small tolerance values this area is negligible and the frames are almost encoded as still images whereas for large tolerance values only a small portion of the frame has to be encoded.

As we have seen the distribution and use of the first, original frame only influences the quality of the first few frames. It is therefore necessary to distribute one original frame in a regular basis in order to keep the quality of the whole video signal at



(a) Tolerance 4 *(b) Tolerance 16* *(c) Tolerance 50*

(d) Tolerance 4 *(e) Tolerance 16* *(f) Tolerance 30*

Figure 5.10 : *Frame 15 of Miss America and Trevor Sequence for different temporal tolerance settings. The white area is covered by static range blocks.*

acceptable standards. However the latter may affect the efficiency of a distributed video system especially when we deal with images of larger size or colour images which require a large number of bits for their representation.

5.3.1. Compressing Every Tenth Frame as Still Image.

In order to overcome the problem of the significant loss of image quality after a few frames a scheme which compresses as a still image every tenth frame including the first one, was adopted. This scheme is inspired by the MPEG standard and aims to limit the transferring of the produced artifacts from frame to frame through the static range blocks.

The first frame is compressed as a still image because the quality of this frame will influence the resulting quality of the following ten frames. Doing this allows a uniform implementation of the algorithm and it makes the resulting signal completely independent to the original one since there is no need to distribute any original frame. However it is now necessary to apply several iterations on an arbitrary image in order to successfully reconstruct the first frame which is very important for the quality of the following ten frames. Since this process is time consuming we used a hybrid scheme. In this scheme the first frame was decoded by an hierarchical method while the remaining frames were decoded by simply applying the IFS code on the predecessor frame, once.

As a result of this approach, the total quality of the video signal is significantly improved for both Trevor and Miss America sequences as can be seen in figures 5.11 and 5.12 respectively. The intensive decline observed on the previous scheme after the first few frames is now missing. Instead we can observe some steep fluctuations especially for higher tolerance values. This is because the technique that was used to detect the motion is naive and at high tolerance values it can not always provide

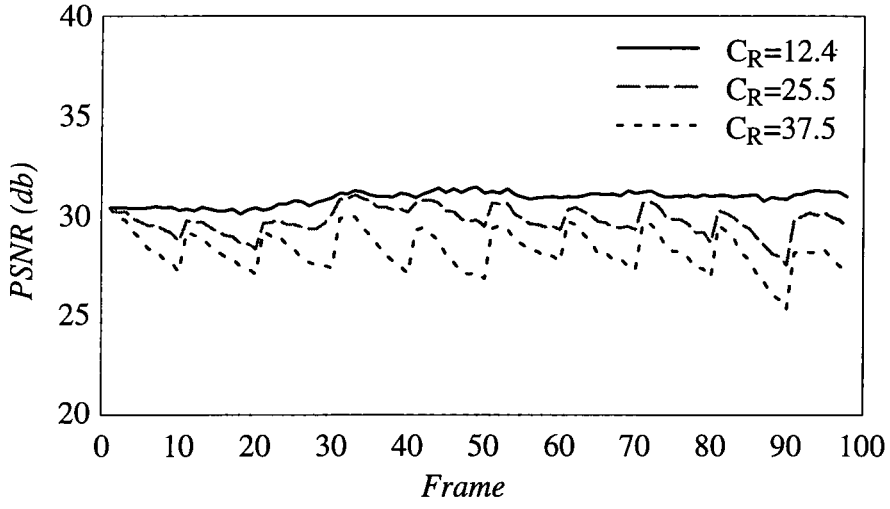


Figure 5.11: *PSNR fluctuation by frame of the Trevor sequence for different compression ratios. Every tenth frame is compressed as a still image.*

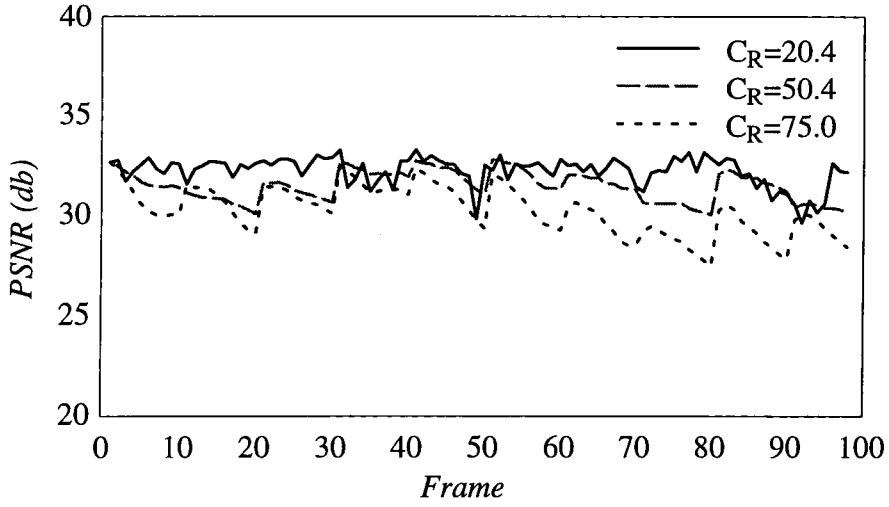


Figure 5.12: *PSNR fluctuation by frame of Miss America sequence for different compression ratios. Every tenth frame is compressed as a still image.*

satisfactory motion detection, resulting at a steep decrease of image quality immediately after the frame that was fully compressed.

Furthermore, in figures 5.11 and 5.12 there are some points where the fluctuations become even steeper and the recovery of the quality when we have a fully compressed frame is not so complete. This phenomenon is due to the sudden movement of the talking head at points which result in a large difference image between the consecutive frames. Therefore, it might be inadequate to apply the transformations only once on the previous frame, in order to reconstruct the next one at high quality.

Another point that has to be discussed regards the execution time for both the compression and decompression processes. In figures 5.13 and 5.14 the curves of the compression and decompression time, versus the temporal tolerance, can be seen and compared with the corresponding curves of the previous scheme as shown in figures 5.6 and 5.7. It is obvious that both processes need more time to execute but this increase is not too significant when the improvement in signal quality is taken into account, as shown in figure 5.15.

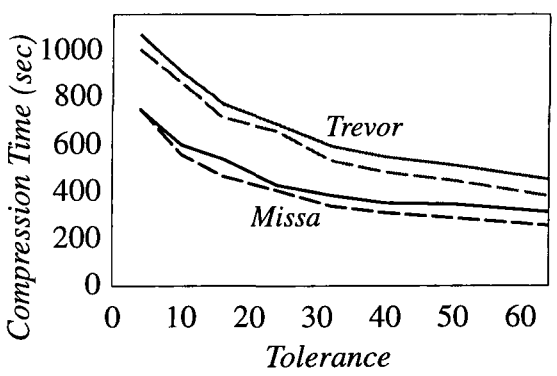


Figure 5.13: *Compression Time versus Tolerance when compressing as still images every tenth frame. The dashed lines show the results of the previous scheme.*

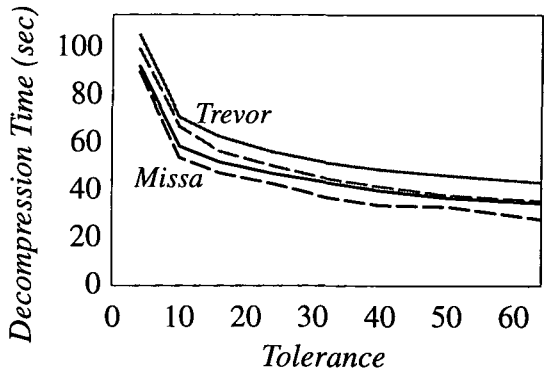


Figure 5.14: *Decompression Time versus Tolerance when compressing as still images every tenth frame. The dashed lines show the results of the previous scheme.*

Furthermore, although figure 5.16 appears to show that the compression ratio is smaller than before, this is not the case. In practice the uncompressed first original frame has to be included in the decompressed signal of the previous scheme (narrow dashes) which decreases significantly the compression ratio depending on the total number of frames.

It can be concluded then that this scheme considerably improves the efficiency of the fractal video compressor and decompressor compared to the previous one as the unacceptable artifacts have been now eliminated. However, it still suffers from the drawback

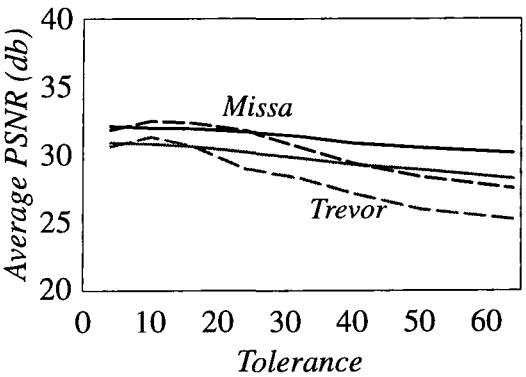


Figure 5.15: *Average PSNR versus Temporal Tolerance when compressing as still images every tenth frame. The dashed lines show the results of the previous scheme.*

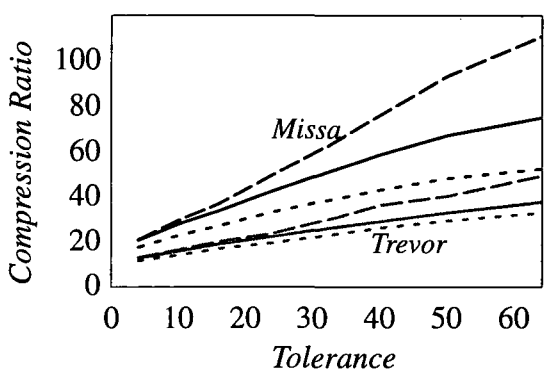


Figure 5.16: *Compression Ratio versus Temporal Tolerance when compressing as still images every tenth frame. The dashed lines show the results of the previous scheme.*

of relatively high compression and decompression times. Another disadvantage is that these times are not the same for every frame since some of the frames are fully compressed. This results on a non continues video signal which pauses at intervals (equal to the number of the partially encoded frames) between two consecutive fully compressed ones (currently 10).

5.4. Encoding Video Frames with Regard to the Encoded Predecessor

An alternative scheme to that described in the previous section was examined in order to provide more efficient fractal video compression and decompression and to eliminate the drawbacks described above. This scheme is similar to the previous one but instead of comparing the range blocks at each frame with ranges or domains in its original predecessor it makes the comparisons needed with the previously encoded frame.

We will now consider this in more detail. Assume that f_i represents the ith original frame and g_i the corresponding encoded frame such that $g_i \approx f_i$. Then, the frame f_i is quadtree partitioned into range blocks of different sizes, ranging from 16x16 to 4x4. Each of these range blocks are first compared with the corresponding blocks in the previously encoded frame g_{i-1} which have the same size with the ranges and sit in the same x and y position.

For the comparisons, the Mean Square Error as defined in equation (5.1) is used. If this is below a predefined temporal tolerance value then the range block is characterized as a static block and it is copied from the g_{i-1} frame. Assuming that the affine transformation has the general form:

$$w(f(x)) = s(x)f(m(x)) + o(x)$$

the latter is equivalent with the above affine transformation, if we take the scaling parameter to be $s(x)=1$ and the offset parameter $o(x)=0$. The corresponding blocks in

the previously encoded g_{i-1} frame can be considered as domain blocks of the same with the range block size, and at position $m(x)=x$.

If on the other hand the Mean Square Error is found to be above a certain temporal tolerance the range block is encoded in the standard way using domain blocks from the g_{i-1} frame which are twice the size of the range blocks to find the best match. In order to reduce the search both the ranges and the domains are classified into 72 categories using the classification scheme described in the previous chapter.

The mapping of static range blocks with domain blocks of the same size might seem to contradict the contraction theorem. However, since each image is encoded in terms of its encoded predecessor, there is no need for the contractivity condition. Using contractive transformations will only result in reduced errors because the artifacts of the previous frame will be reduced in size[33].

For instance, suppose that the union of these mappings define an IFS. Then each affine transformation satisfies the following relation: $w_i(f_{i-1}) \approx f_i$ where w_i is now a more general transformation which includes motion compensation and some type of spatial encoding, f_i is the *ith* frame of the original sequence and $f_0=0$. Assuming every w_i is contractive then the contraction theorem states that there is a positive number $s<1$ such that:

$$d(w_i(f), w_i(h)) < s\ d(f,h) \quad \forall f,h \in \mathbb{R}^2$$

where d defines a metric. In addition, if g,h are encoded image functions they will satisfy the equation $w_i(g_{i-1})=g_i \approx f_i$ and $g_0=f_0$. Considering all the above the following can be extracted [33]:

$$\begin{aligned} d(g_i, h_i) &= d\big(w_i(g_{i-1}), w_i(h_{i-1})\big) \\ &< s\ d\big(g_{i-1}, h_{i-1}\big) \\ &< s^i\ d\big(g_0, h_0\big) \end{aligned}$$

From this we can see that if $i \rightarrow \infty$ then $d(g_i,h_i) \rightarrow 0$, which means that the errors on the frames will disappear when the transformations are contractive.

Based on this latter property the problem of the first frame can be ignored and it may reconstructed by applying the transformations only once. This results on a first frame of low quality and blocking effects. However, the initial problem of blocking will disappear rapidly in the following frames since domain blocks that are twice the size of the range blocks are used to map the dynamic ranges. This phenomenon can be seen in figure 5.17 for Miss America sequence. For large temporal tolerance settings the elimination of the initial blockness is slower than for smaller settings. The latter can be observed in figures 5.18 and 5.19 which shows the PSNR fluctuation for different compression ratios. The image quality is low for the first few frames but it soon improves up to an almost constant value which depends on the compression ratio. Therefore it is not necessary any more to fully compress frames in a regular basis as we did with the previous scheme. The process of the encoding and decoding needs almost the same time to be implemented for every frame. The decoding time is only slightly influenced if there is sudden motion.

(a) first frame *(b) second frame* *(c) third frame*

Figure 5.17: *The reconstructed first three frame of the Miss America sequence. The elimination of the initial blockness is obvious.*

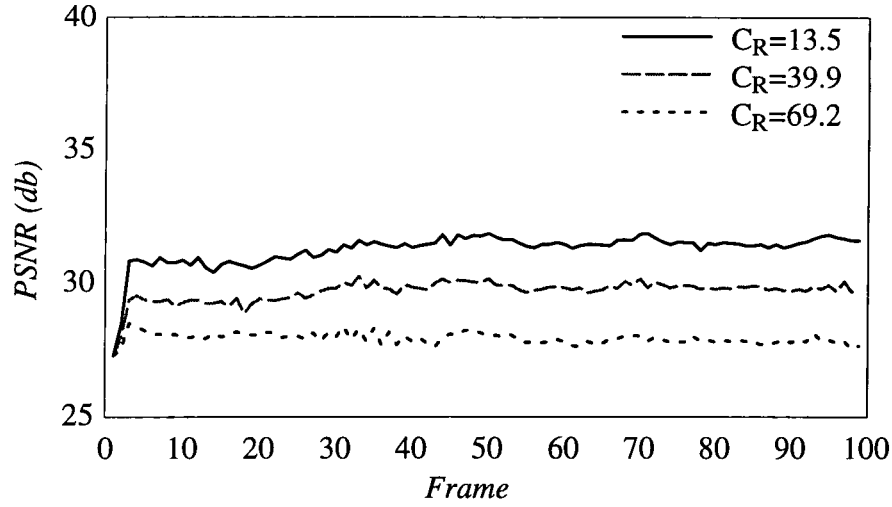


Figure 5.18: *PSNR fluctuation by frame of the Trevor sequence for different compression ratios. We consider comparisons with the previous encoded frame.*

In practice there are many significant advantages of this scheme. The increase of the image quality due to the minimisation of the errors in the previous frames allows us to set even larger temporal values thus improving both the compression ratio and the encoding and decoding time. In figures 5.20, 5.21 and 5.22 the variation of the average PSNR, the compression time and the decompression time with respect to the compres-

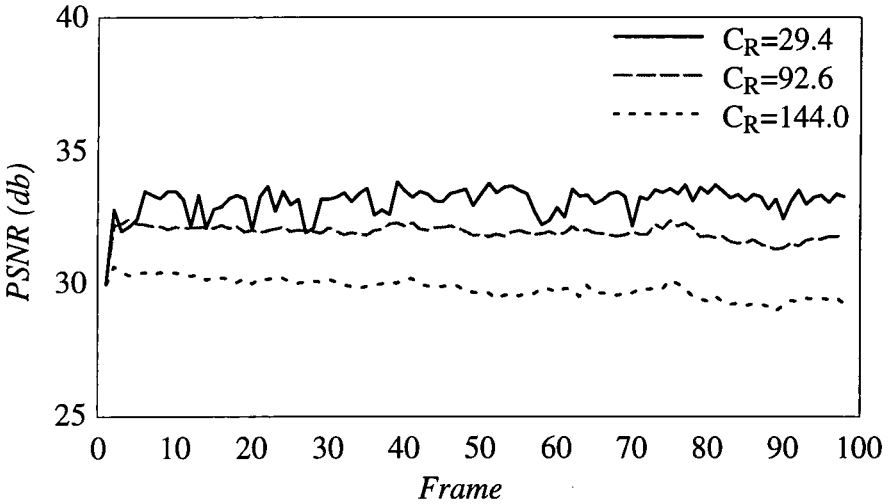


Figure 5.19: *PSNR fluctuation by frame of the Miss America sequence for different compression ratios. We consider comparisons with the previous encoded frame.*

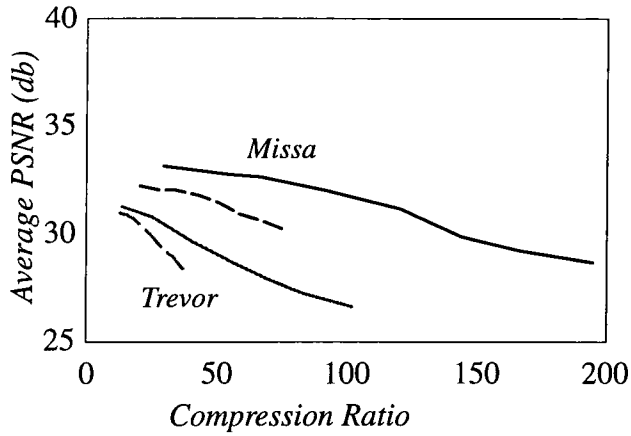


Figure 5.20: *Average PSNR versus Compression Ratio when compressing with respect to the previous encoded frame. The dashed lines show the results of the previous scheme.*

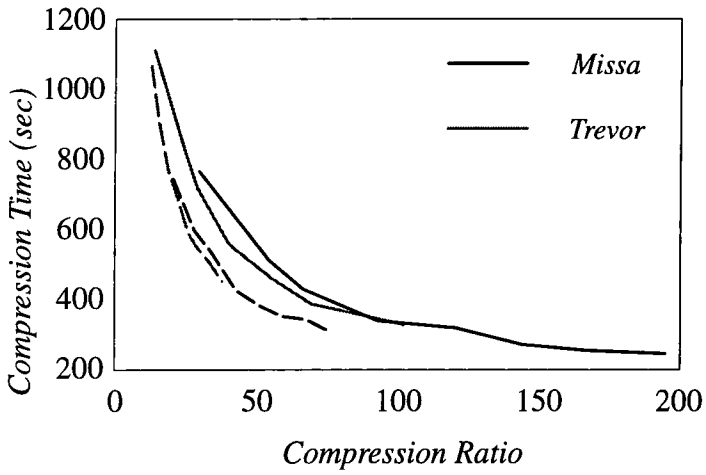


Figure 5.21: *Compression Time versus Compression Ratio when compressing with respect to the previous encoded frame. The dashed lines show the results of the previous scheme.*

sion ratio are shown respectively and compared with the corresponding variations of the previous scheme (dashed lines).

In particular, figure 5.20 shows that the decrease of the image quality for larger compression ratios is not as steep as before. Therefore compression ratios of over 200:1 can be achieved for acceptable image quality. On the other hand figures 5.21 and 5.22 show that the compression and decompression times are larger for small compression ratios. However faster implementation of the algorithms can be achieved in fact, if we consider higher compression ratios. For example, on a modest workstation, we achieved

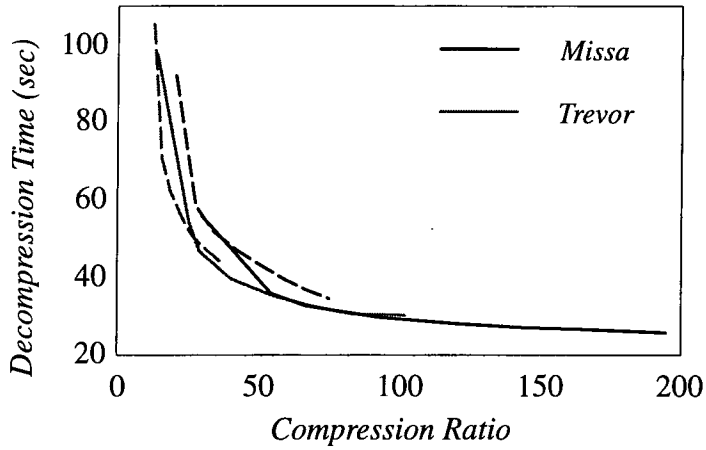


Figure 5.22: *Decompression Time versus Compression Ratio when compressing with respect to the previous encoded frame. The dashed lines show the results of the previous scheme.*

almost 4 frames per second for the decompression and 2.5 seconds per frame for the compression.

The above scheme is similar to the one accredited to Fisher et al[33]. Fisher's scheme also encodes each frame using image peaces from its encoded predecessor. The difference is that instead of searching static areas, it considers a pool of domains on the previously encoded frame which have the same size with the range block. Then his algorithm tries to find an appropriate transformation by searching throughout this domain pool, the best range–domain mapping.

5.5. Summary

In order to efficiently implement fractal compression techniques on a video signal the temporal redundancy has to be removed. Compression schemes that implement three–dimensional fractal compression have been described and found to be inappropriate for a real–time video distribution system since they handle many video frames at once and the decoding process requires all of these frames to be considered simultaneously.

Three different schemes which extend the still fractal compression techniques to include the temporal redundancy were investigated. All of them were based in the quadtree partitioning of each frame into ranges and the characterization of these blocks as static and dynamic, depending on the difference of this block with its corresponding predecessor. The main difference between these schemes was the way in which they handled the first frame, which has no predecessor. In the first scheme, the original frame was considered while the second one implemented a full encoding and decoding on an arbitrary frame. In addition in this scheme, implementation of a full compression in a regular basis (every 10 frames) was used, in order to improve the quality of the signal which was contained many artifacts in the first scheme.

In the last scheme, the comparison of each frame with its encoded predecessor was considered. Although it may appear that this will increase the execution time of the encoding since it requires that the frames are decoded at the same time, in practice the scheme performs much better than the previous ones, as far as the encoding and decoding time needed and the resulting image quality for the same compression ratios. However, this performance is still some distance from a real–time implementation on the SUN microsystem IPC station which was used for our experiments. The decompression process has to be speeded up only by a factor of 6 which is surely achievable with an improved workstation, while the encoding has to be speeded up almost 70 times to achieve real–time execution. Therefore more optimal algorithms for fractal compression and other hardware topologies (i.e. parallel DSPs) have to be considered in order to integrate a video conferencing system based on the fractal block coding method.

Chapter 6

DATA DISTRIBUTION AND NETWORKS

We have already investigated several techniques for implementing fractal video compression and examined ways for improving its efficiency in terms of execution time, image quality and compression ratio. The need for efficient distribution of the compressed signal now arises, since this is an essential parameter of a video distribution system. We will therefore introduce the basic concepts of data communication and networking principles by defining the terminology and examining several network topologies, in order to identify the advantages and disadvantages of each telecommunication system.

6.1. Information and Communication

Distributed computing systems are generally dealing with two different terms, these being the information and the communication. *Information* is a concept which is quite intuitive for most people. It is a fundamental quantity which can be transmitted using a small amount of energy. The most valuable property of information is that it can be duplicated without degrading. The video or audio signals and the text documents are some examples of this quantity.

Communication on the other hand is defined as information moving in space. It is an essential process in all organized systems. As our society is developing, the communication is becoming more and more rich thanks to the introduction of technological infrastructures which allowed, for example, the development of the telephone network or the introduction of the television and other telecommunication systems.

All communication processes need a *sender* and a *receiver*. The sender transmits a message (which is a sequence of information symbols) to the receiver, along a channel or medium, which is common for both of them. It is necessary for the sender and the receiver to use the same code for transmitting the message along the communication channel and extracting the information from the message.

A factor that is always present and influences negatively the transmission process, is noise. Sometimes it can be negligible, but if the signal to noise ratio is small, the incoming messages can be distorted and become intelligible, or can be interpreted in a wrong way by the receiver. Since the primary aim in data communication is to achieve reliable message transfers, at a given time, from location A to location B, several transmission techniques are used to protect the signal from the noise, validate the incoming messages and recover the corrupted messages. Usually it is not economically feasible to completely eliminate the noise from a channel, but fortunately there are ways to cope with it and reduce it to an acceptable level which almost do not interfere with the communication process.

The model for data transmission is quite complex as several factors are involved. It is possible to select for instance, the physical configurations of the channel (such as point–to–point, multipoint, star, ring, e.t.c) or the media used (wire, satellite, optical fibre). Furthermore the rules used for coding the message and the *protocols*, (which are procedures for error detection and recovery, flow control and management) have to be defined.

Generally speaking, when a communication system is designed it is necessary to consider three different components which are involved in the transmission: the hardware used, the processing software and the structure of the messages.

6.2. Transmission media

One of the most important hardware component on a data communication system, is the transmission media. As transmission media, the following can be used [47] [49] [50]:

- Twisted pair: It consists of two insulated conductors which are twisted together. It is the least expensive solution, but it has a limited transmission rate and it is susceptible to interferences.
- Coaxial cable: It surrounds the inner conductor with a dielectric, and a coaxial tube of solid or braided metal surrounds the dielectric. Electrical interference is extremely low if the outer shield does not have gaps. Typically a bandwidth of 10 – 20 Mbits per second over several hundred meters of coaxial cable can be achieved.
- Optical fibres: They transfer light waves instead of electrical signals. Fibre optic cable cores are made from glass or plastic. They have the higher transmission speed and capacity. They have no problems correlated with electromagnetic interference, thus, they provide very low error rates. They are general smaller and more flexible than electrical cables. The cost, when medium and long distance fibre optic–links are considered, is less than the one for electrical cables.
- Satellites: Instead of using a physical line for the transmission, electromagnetic waves can be used through free space. For instance a microwave beam is directed to the satellite, and the satellite redirects it towards another place. It is usually used for long distance data transmissions or when it is impractical to use a physical line.

The most widely used media, even today, is the coaxial cable. However the optical fibres are getting more and more popular, especially when we deal with Local Area Networks.

Video distribution systems involve the communication of devices in both local and wide area networks. Therefore, we will present these networks in more detail, in order to acquire a global view of the various network topologies.

6.3. Local area networks

Local area networks (LAN) allow a group of independent computers, spread over a local geographic area (normally less than 1 Km), to share information and resources via interconnection, which allows high transmission and low error rate. LANs are serial bus systems. Some of the elements which qualify a local area network are *the network topology, the transmission techniques* and *the medium access methods.*

6.3.1. Network topology

The topology of the LAN refers to the way in which the computer and devices are connected to the physical cable. Today three forms of topology are used for communication networks[2][47][50]:

- *Star network*: This topology requires that each system is interconnected with a central station or hub which controls the flow of information between the elements of the network. It is usually used to interconnect local terminals to mainframe computers. If the central hub fails it produces the complete breakdown of the network. Single cable failures affect only single stations. The installation of new elements in the star system requires an intervention on the hub which is not always possible.

- *Bus network*: This solution employs a single cable to interconnect all the systems. The communication between two systems can be established without involving a central controlling station. The total throughput capability generally decreases as the number of stations increases. New stations can be added without reconfiguration of

the network until the performance of the network is acceptable. The failure of a single device (if a central system is not used) should not prevent the other systems from communicating. The disadvantage of this topology is that failure of the bus cable makes communication between nodes impossible.

- *Ring networks*: This topology creates a loop by connecting each system or device to its neighbour. The interface which is part of the system is responsible of relaying the information around the ring. The number of nodes is limited to by the system design, and any additional node produces system disruption and reduces performances. A ring is vulnerable to a single break in any link or node repeater.

6.3.2. Transmission techniques

In order to allow more stations to share a communication channel two techniques are used [47]:

- *Time division multiplexing (TDM):* Using this technique, the channel is divided in periodic time slots and users can only transfer data on the bus only one after another, making rules for bus access necessary. This method can be easily implemented and it is the most frequently used in network busses.

- *Frequency division multiplexing (FDM):* In this case, the channel bandwidth is divided into frequency bands, each allocated to one virtual channel.

The two types of multiplexing are equivalent in practice as they require that the data is processed at both ends of the transmission channel. However TDM does not require high frequency equipment, and if a user do not need the channel, its empty slot can be allocated to someone else.

6.3.3. Medium access methods

Medium access is controlled according to rules which every station on the bus should observe for transmitting and receiving messages. The access methods which are usually employed in LANs are [46][50]:

- *Polling:* If the polling technique is used, a master station interrogates sequentially the nodes connected to the network giving the access right to the transmitting media to the stations which need to transmit. The weak point of this architecture is the polling station; if it breaks down all the system fails.

- *Random access:* In the random access technique there is no master station, and the access of the medium is determined by chance. When a station wants to transmit a message it waits until the transmitting medium is free from traffic and it sends its message. If another station starts the transmission at the same time a collision is generated and the conflict is resolved by different algorithms. Ethernet is the best known example of application of this method.

- *Token passing:* No master station exists when a token passing technique is used. A token is circulated among the nodes which retains it for up to a stated interval of time. Token ring and token bus are two examples of token passing networks.

6.4. LAN Standard Protocols

The diverse LAN approaches have begun to settle down into standardized categories. The most comprehensive documentation and specification of LAN has been defined by the professional engineering society IEEE which has published a set of documents known as "IEEE 802" standard, which describe both general principles and particular types of LANs. These documents have been adopted by ISO as Standard ISO 8802 .

The committee intended to create a unique standard for local area networks, but there were three major technologies mutually incompatible supported by powerful companies.

As no agreement was reached among the parties supporting the different technologies different standards were developed. They are:

- IEEE 802.3 (Ethernet);
- IEEE 802.4 (Token Bus);
- IEEE 802.5 (Token Ring).

An overview about the main characteristics of theses standards follows.

6.4.1. IEEE 802.3 (Ethernet)

Ethernet is a widely used local area network for both industrial and office applications. It uses a serial bus topology with random access and a protocol called CSMA/CD (Carrier–sensing Multiple Access/Collision Detection) for the medium access coordination. If a station wants to send a message it waits until there is no traffic on the bus and then transmits its information along the communication channel. If another station does the same a collision is generated and both stations detecting that their signals are garbed stop transmitting (see fig 6.1). After every collision each of the stations involved in the event use an exponential backoff algorithm which randomly decides when each system can try again to use the bus to complete its activity. If on the new attempt of gaining control of the bus another collision is detected the waiting time is doubled. If a station is not able to gain access to the bus after a certain number of attempts it assumes that the problem has a different nature and reports the situation to the higher layer [46].

The data transmission rate is 1, 10, 20 and 100 Mbit/s, whereas the maximum length between the most remotely connected points is about 2500 m. Ethernet architecture is flexible and open. There are a lot of suppliers of Ethernet devices. The medium used is coaxial cable and it has not active part in the network control.

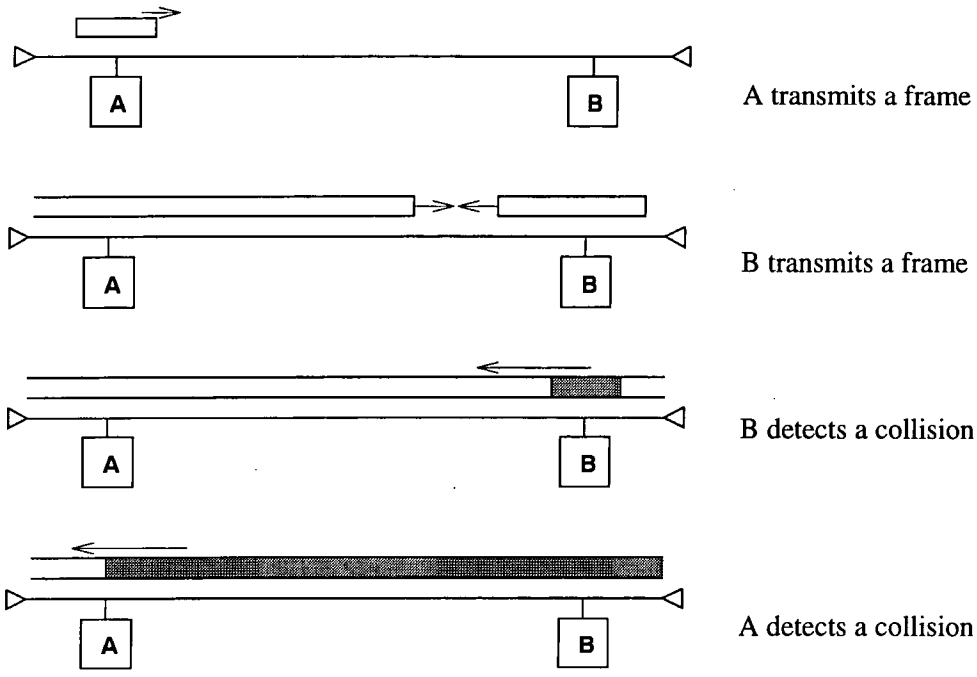


Figure 6.1: *Ethernet operating principle[48].*

A disadvantage of this type of approach is that when the load of the network increases collisions are more frequent degrading the performance of the network[46].

6.4.2. IEEE 802.4 (Token Bus)

Token bus was developed for factory automation activities. All the nodes are connected to the same bus. It uses a token passing technique for accessing the communication channel. Only one unit at the time can use the bus for transmitting messages. A node is entitled to forward data over the medium only if it retains the token, which allow the holder to transmit up to a specified amount of time. When a station has completed its transmissions or the time at its disposal is expired it passes the token to the next station in the logical ring, thus each station has the opportunity of transmitting (see figure 6.2). The way in which the token is passed makes the token bus a logical ring. If the token is lost one station, which is in charge of the task, will replace it [46].

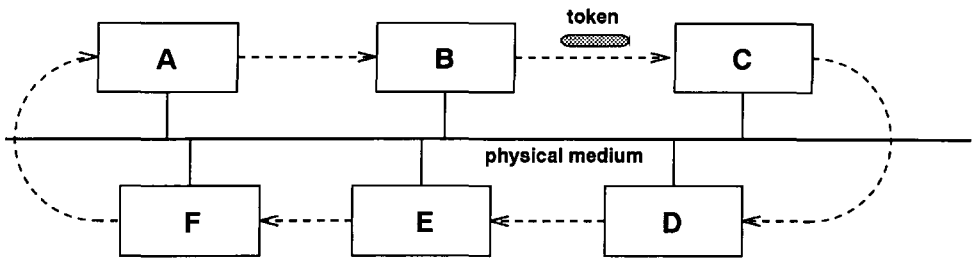


Figure 6.2: Token bus operating principle. The stations A–F circulate the token and can communicate when they hold it.

This system does not allow the possibility of collisions as the access to the bus is granted by the possession of the token. If a new node is added to the bus it is necessary a reconfiguration of the system to include the new station in token passing mechanism.

The data rate is in the range 1, 10 Mbit/s. The worst–case delay for a node to gain access to the bus is computable, and can be designed to meet the real–time requirements. The increasing of the number of stations on the bus decreases the performance of the system [50].

6.4.3. IEEE 802.5 (Token ring)

Token Ring was introduced by IBM. The nodes are connected together in a ring topology and the operating principle is similar to the one used for the Token Bus (see figure 6.3). The token is circulated and each station receives it; the station holds the token if it wants to transmit a message. the ring transmits data in only one direction. The message is sent by passing it onto the ring instead of the token or can be appended to the token itself. The message is passed from one station to the next one until it reaches the destination node where it is retained. The successor and the predecessor of a node are uniquely chosen by their physical arrangement within the ring [46][50].

For introducing a new node the system has to be shut down, in order to connect the new station in the rink topology. The data rate is of 1 and 4 Mbit/s. Even in this case the worst–

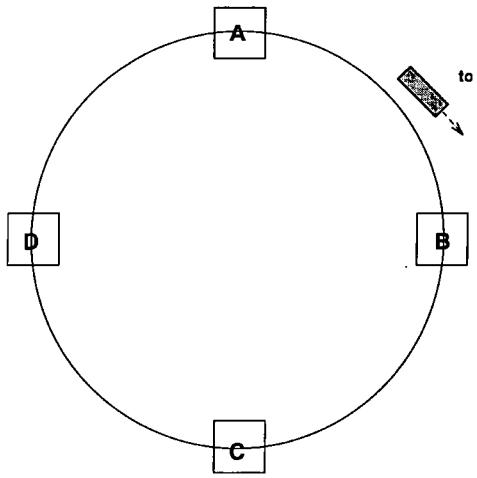


Figure 6.3: *Token ring operating principle. The token is circulated among the stations. A station can send messages only when it holds the token. The stations connected identify and collect the messages directed to them and forward the others.*

case delay can be calculated, but it is generally greater than the one produced by a Token Bus system with similar characteristics.

6.4.4. Comparisons

A comparison among the different LAN systems defined in the previous sections can underlying the advantages and the disadvantages of the single approaches [46]. As the Token Ring and Token Bus have quite similar properties they are treated together.

- ***CSMA/CD (Ethernet):*** It has a very good performance in lightly loaded, random, environment. As there is no medium access control the implementation of the system results simplified, and there is no need for monitor or token requirement functions. Stations can be easily added or deleted simply activation or deactivating them. Stations operate following the principle "every user for itself". A station need only to wait for the bus to be available: once a frame is sent it can continue using the bus until a collision is detected. A station throughput depends only on the activity of the other stations when it attempts to transmit, while the number of the connected stations in not important. When the load increases a delay in completing operation appears due

to the greater number of collisions on the bus. As the behaviour of Ethernet is not predictable on a small time scale, and therefore it is not possible to guarantee a certain level of performance. A minimum frame length is required to assure accurate collision detection. A fault on one of the stations do not compromise the functionality of the network. Only the bus itself is the weak point of the system and if it is interrupted all the system fails.

- ***Token Approaches:*** These have completely different properties in comparison with the CSMA/CD. The token passing protocols require transmission capacity and time delays for protocol related information exchange. They are more expensive than the Ethernet solutions as they have to generate and process medium access control protocol data units. Token Passing requires additional procedures for adding and deleting nodes. In the Token Bus it is necessary to add or delete via software the involved stations form the logical ring. In the Token Ring if a new station is introduced the network had to be disabled to insert the new node in the ring. It is possible to predict the response time, and the token holding time can be adjusted to tune the performance. Token passing performance is a function of the number of active participants. Increasing the number of stations involved in the passing of the token has as a result the increase of the delay and the reduction of the throughput for everyone. The station throughput is directly related to the rate at which it can obtain the token. In the ring the token returns after traversing all the connected systems. The minimum waiting time is equal to the sum of the propagation delay and processing delay of all the other system, while the maximum interval is equal to the sum of all the time interval used by the other system to transfer their information. Finally, in the Token Bus, if the logical order do not resemble the physical order of the nodes, propagation delays can grow very quickly. In both systems if there is a fault in a single node all the systems fails.

6.5. Wide Area Networks

The Wide Area Networks (WAN) are networks serving at a large geographic area varying from a city to an international zone. They are used as a backbone to interconnect autonomous local as well as smaller wide area networks. The WANs are generally based on transmission lines which interconnect switching nodes using several transfer mediums such as satellites and leased or analogue telephone lines. They can be very tolerant to hardware and software failures and thus very reliable.

The Internet can be considered an example of WAN. In fact it is the ultimate WAN since it consists of many networks all over the world, built in a hierarchy. These include the ARPA network, NSF network as well as sections of the JANET and other military and academic WANs and LANs.

6.5.1. Connecting to the WAN.

The main drawback of the WAN technology is the data transfer rates available. A typical connection to a WAN which uses modems[2] and transfers analogue signal through the traditional telephone network (circuit switched), may operate at transfer rates of up to 28800 bits per second which is far from the competition in the LAN technologies. However, this situation is changing, as digital service WAN technologies are improving and becoming more and more faster and dynamic. The dominant technologies in the digital connection are the following[53][54][55]:

- *ISDN (Integrated Services Digital Network)* which integrates voice (telephony) and data (i.e. packet switching) into one unified line. It is a multiple–channel service composed of three different types of information channels into a single physical set of wires (the B channel which operates at 64Kbps, the D channel at 16Kbps and H channel for much higher transfer rates). The ISDN basic interface has been designed so

2. MOdulator –DEModulator

that existing telephone wiring will be able to support the new digital service (it has a built–in packet capability). It promises fast line speed since since it can achieve up to 128Kbits per second transfer rate and high line quality.

- *ATM (Asynchronous Transfer Mode)* service is a less commercialized technology which handles traditional as well as multimedia data at throughput speeds that scale from 50 Mbps to 622 Mbps. In ATM, data whether it be voice, video or anything else is broken up into packets of 48 bytes plus 5 bytes dedicated to the protocol. Each packet carries the destination address and is free to choose the path across the WAN that is faster to the intended receiver. When all the packets arrive the receiver reassembles the data into its original form. Although the ATM is a promising WAN technology which can efficiently link remote sites via videoconferencing, however it still is on its early stage of development and a few more years will be needed before its implementation on commercial hardware.

Despite the performance advantages of the above digital technologies it is not always possible to obtain small price/performance ratios, therefore the conventional technologies are still used more usually.

6.6. Network Architectures and Protocols

After discussing the various types of data communication systems it is necessary to define the network architectures (protocol architectures) and outline the different types of communication protocols that are used in distributed programming. A network architecture is the set of rules which govern the connection and the interaction of the network components. It includes the data format, protocols and logical structures for the functions which provide effective communication between data processing systems connected to the network [47]. The layering activity of the transmission system separates the functions into distinct levels which can communicate individually.

The existing communication systems can be classified into the following architectures [49][48]:

- The Closed System Interconnections (CSI). These are local networks in which all the components come from and are designed by the same vendor. The user is therefore forced to buy complete solutions and future extensions from one vendor. If devices of different vendors have to be connected, special solutions, usually expensive and complicated, have to be customized.

- The Open System Interconnections (OSI). In order to overcome the incompatibility problem and give some structure to the area where digital transmissions are performed the *Open System Interconnection* has been introduced. The OSI architectures try to develop vendor independent architectures, allowing communication between application systems which are usually not compatible with one another. This requires extensive standardization work which concerns the communication interface and the way in which the messages are exchanged.

6.6.1. The ISO / OSI model

In 1983 the International Organization for standards defined its Open System Interconnection ISO / OSI reference model (ISO 7498) [51]. OSI itself is not a standard but offers a framework to identify and separate the different conceptual parts of the communication process. It introduces a conceptual model for communication which is similar to the different levels of operating systems, where the operations have different abstractions, ranging from machine code and assembler programming to high level languages and applications. The multi–layered structure of the communication process is essential for providing services while hiding their implementation details from the higher layers.

Seven functional layers are defined in OSI. Every layer provides services to the next higher layer and uses services to the next lower layer to execute its task. Therefore each

layer exchanges information only with the adjacent ones [50]. OSI service calls are similar to operating system calls: the requesting layer passes data and parameters to the layer below and it waits for an answer ignoring the details of how the request is carried out [46].

The layered approach ensures modularity and the facility for networking software to be upgraded without affecting the other layers of the communication system. The modularity, as already mentioned, makes it possible to use multi–vendor hardware and software in the same system.

Modules located at the same layer but in different position in the network are called *peers*; they communicate via *protocols* that define conventions for exchanging information between two users of the same information. The services (what to do) are strictly separated from the protocols (the actual implementation). The layers defined in OSI are the following[48][49][50]:

- *The physical link layer*: this layer is the lowest layer of the model and is concerned with all aspects of the physical interconnection of the interface to the cable. For example the standards will define the mechanical aspects such as the type of connector to be used, the electrical signals levels used for transmission and reception via the cable, and functional and procedural aspects such as the type of handshaking to be used. The reference model does not extend down to the physical medium itself so does not include a specification of the type of cable to be used. The physical layer is the only real connection between two communicating nodes.

- *The data link layer*: this layer provides the functional and procedural means to establish, maintain and interrupt a data link over the network. It defines the way the serial data is organized and the detection and possibly correction of errors occurring during the packet transmission.

- *The network layer*: this layer has the task to do the routing, the selection of a path through a network node, on which data transfer takes place. As the paths should be never overloaded the most efficient path must be found.

- *The transport layer*: this layer provides an end–to–end communication control and it is the interface between the application software that requests dàta communication, and the external network. It has the responsibility of verifying that the data from one machine to another is transmitted and received correctly.

- *The session layer*: this layer is involved with establishing the interactions between two users applications on different systems connected to each other by the network.

- *The presentation layer*: the main function of this layer is to provide an independence to the user application from differences in the presentation of data. Thus differences in the way that computers talk to one another and the actual data can be resolved.

- *The application layer*: this layer provides application specific protocols. It provides a set of services which can be directly called by application programs.

The first four layers are called *network oriented protocols* whereas the remaining three are called *application oriented protocols*. As data moves downwards to the physical layer two major things happen to it. Additional protocol–related information is appended to the original message and the latter may be segmented into smaller pieces (packets)[46].

The OSI standards appear to be one of the leading forces for the future of the inter–networking. However OSI model is not exempt from criticism. Efficient implementation has not been explicitly addressed at all. Moreover some aspects of the implementation may require to use excessive resources of the system, and may produce performance degradation. Several times has been pointed out that the division of the layers 4–7 is somehow academic. Even if there is the implicit requirement that all layers should be involved in each application process activity, OSI in usually not implemented in its entirety, but computer manufac-

turers implement only the necessary layers avoiding all the features that are not necessary for the specific application. The most important faults in the OSI system is that it does not include important functions as network management and security (data encryption): these omissions may cause in future the development of incompatible standards [46].

6.6.2. Alternative Network Architectures: TCP / IP

While most manufacturer of computer communications systems and equipment vendors are evolving products following the directives provided by the ISO/OSI model, there are a variety of alternative network architectures and protocol layers currently in use. It is unlikely that in the next near future all of them will be abandoned, but in many cases it will be necessary to provide gateways between systems which comply with the ISO or other standards. Amongst others, alternative network architectures include:

- The Systems Network Architecture (SNA) by IBM.
- The Distributed Network Architecture (DNA) by DEC.
- The Xerox Network Systems (XNS) by Xerox.
- The Transmission Control Protocol / Internet Protocol (TCP/IP) by DARPA.

The most important of the alternative architecture is TCP/IP. Its developing started from the early '70s by the US Department of Defense Advanced Research Projects Agency (DARPA), and it was included in the Berkeley Unix version 4.2 operating system [1]. The Unix–TCP/IP is becoming the most widely spread system used in local as well as in wide area networks for interconnecting computers together. Although its aims are similar to the OSI, it however provides a smaller level of functionality then the latter. The main advantage of this architecture is that it is non–proprietary, and due to their popularity there is abundance of software written that use these protocols.

In figure 6.4 a comparison between the TCP/IP layers and the OSI ones is shown. The bottom layer includes in fact the physical and data link layer and it is called network access layer. It embraces those protocols, such as the X.25 protocol, that are necessary to interface to the network being used whatever the medium may be (Ethernet, fibre–optic cable or satellite link).

The OSI network layer corresponds to the *Internet Protocol (IP)*; this protocol addresses messages, routes them across the network and exchanges data between systems independent of the network topology and the media used. It is a connectionless protocol which means that it handles every data package separately. The IP protocol uses for the routing of the messages not a physical address as in the previous layer but instead is a 32–bit number which is written as four decimal 8–bit numbers (i.e. 129.234.207.112) and is called the internet or IP address. The data transmission at this level is not reliable. An additional internet control message protocol (ICMP) provides network layer management and control functions.

The OSI transport layer corresponds to the *Transmission Control Protocol (TCP)* which is a connection–oriented protocol. The latter means that several functions are provided for

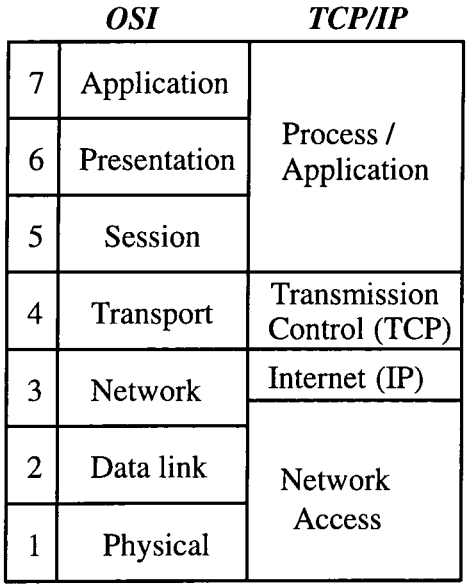


Figure 6.4: *Comparison of the TCP/IP protocol architecture with the OSI one.*

establishing a connection as well as closing it down. It is responsible to ensure that the data arrived on the other end, it keeps track of what has been already sent and retransmits anything required. In order to handle the data more efficiently it breaks it into *datagrams* which are data packages of a convenient for transmission size (normally 128–bytes per datagram). TCP generally ensures the reliable data transmission and the correct sequencing of the data string without repeats.

In fact there is an alternative transport layer in UNIX which is also commonly used, this being the *User Datagram Protocol (UDP)*. In contrast to the TCP this protocol is connectionless, therefore every datagram is handled independently and it is needed for each one of them to include a header which identifies amongst others, the port number of the transport endpoint. In addition UDP preserves the message boundaries but it does not confirm the correct distribution and sequencing of the data string.

In the TCP/IP architecture there are not application–oriented layers. Instead the functions associated with these layers have to be written into the user processes. However a number of application processes have been written to provide specific services, similar to those provided by the specific application service in the OSI application layer. Some of the most commonly used are the following:

- The *Telnet* protocol which corresponds to the session and the presentation layer and allows a user on one machine to log on to another machine by establishing remotely a virtual terminal.

- The *File Transfer Protocol (FTP)* which corresponds roughly to the presentation and the application layer and provides file transfer services between dissimilar machines and operative systems. It supports a variety of transfer modes, file store structures and data types, including ASCII and binary.

- The *Simple Mail Transfer Protocol (SMTP)* or as most commonly referred, electronic mail which allows the user to send and receive messages from specific remote sites.

The TCP/IP architecture assumes that there are many independent networks connected via gateways and the user is able to access network nodes to any of these networks. The Internet is based upon this architecture and it is the above application processes which are commonly used to access it [56].

6.7. Distributed Programming

6.7.1. Sockets

Sockets is a set of system calls which give programmers access to the networking services of TCP/IP and UDP/IP protocols. It is an end–point of communication which fills the gap between the application program and the underlying transport protocol. A pair of sockets can be used to establish client–server communication and manage the flow of data between two machines.

The basic issue of sockets is how they are addressed. There are two different ways, the first one being the so–called UNIX addressing where sockets are named as UNIX path names and the second one the internet addressing. The internet addressing is the one that is most widely used as a data communication mechanism on a network and is mainly consisted of two numbers. The first one is the 32–bit IP address of the host which was described above and the second is a 16–bit port number. The use of a port number is necessary because many socket–type communication processes between the same machines might run concurrently and any of these processes might have many individual connections open.

In general two types of sockets can be created:

- The listening sockets which wait passively for a connection request. These are usually the sockets created to the servers.

- The connection–seeking sockets which are requesting connection to a remote listening socket at a specific host and socket address. These sockets are created to the clients.

The connection can be established between pairs of one socket at the server and one at the client as illustrated in figure 6.5. However after the connection both sockets are handled in the same manner and serve as logical descriptors for file transfers using common read and write commands.

A drawback of the socket programming is that it is needed for a client to know the complete address of the listening socket at the remote server, therefore the port number has to be notified in addition to the machine's IP address. For system programs this drawback is overcome since they have standard, reserved port addresses. For instance the FTP pro-

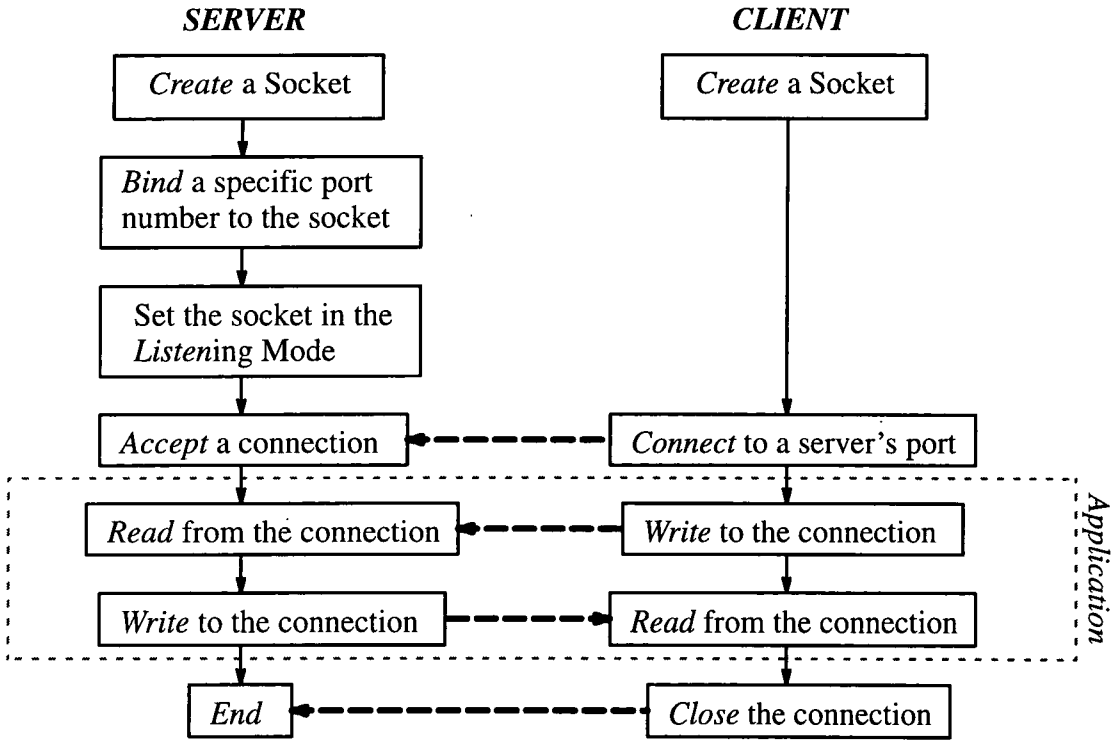


Figure 6.5: *Client – Server Communication using Sockets[1].*

gram will always connect to socket port number 21 and the sendmail server always listens on port 25. When a user is connected to one of these sockets, a child socket is created on a new port number, thus the reserved port number remains free and waits for new connections. However this utility is not provided for user programs. In this case a fixed address has to be used which might be already in use somewhere else in the system

6.7.2. Remote Procedure Calls (RPC)

Alternative methods for writing distributed applications are those using *Remote Procedure Calls (RPC)* or *Transport Level Interface (TLI)*. The TLI offers much the same facilities with the sockets but sockets existed long before TLI and there is already a lot of socket code written around the world. Although sockets do not offer as much flexibility as TLI, they are easier to handle and they are better integrated on the UNIX operating system.

RPC calls on the other hand is a higher–level protocol which uses specialized procedure calls to establish communication and transfer data. In this case, the client calls a local procedure which sends, via the network transport provider, service request messages to a remote machine. These messages include arguments and additional data denoting an existing procedure that needs to be called. When the message arrives to the server, a dispatch routine retrieves the parameters from the message and passes them to the denoted procedure. The data outputed from the procedure is again routed towards the client.

In such way the interprocess communication is established via ordinary function calls, using the same mechanism as it is used for local procedure calls. Only the IP address is needed for the client's RPC calls in order to identify the machine running the server process.

Although it seems that distributed programming using RPCs has nothing to do with sockets, in fact socket–based communication is still taking place beneath the RPC protocol. However the socket port address is not needed to indentify the remote endpoint. Instead

numbers indentifying a program (a group of related procedures), its version and the procedure is all that is needed.

Although RPC have several advantages compared with sockets, it is not always possible to obtain fast interconnection[57] and therefore acceptable performance may only be obtained using sockets.

6.8. Summary

The basic concepts of networks and data distribution where presented in this chapter and the network terminology was defined. We can conclude that since the Internet, the largest and best known WAN is based on the TCP/IP model this is more likely to be used as the protocol architecture for video distribution and conferencing. The TCP/IP model also offers a large number of application processes like the FTP, Telnet, Electronic Mail and a relatively easy, from the programmer's point of view, way to create new applications by establishing communication and transferring data using sockets or RPCs.

On the other hand, having examined several LAN topologies and standards we can conclude that at the bottom level of the TCP/IP model the Ethernet protocol is ~~by no means the~~ dominant and is used in most of the applications. Although this protocol is not excess of judgment it provides a good performance to price ratio. As for the interconnection of the individual users or LANs to a WAN two novel technologies, the ISDN and ATM, which uses digital signals instead of the traditional analogue ones promise real–time implementation of video conferencing through the extremely high bandwidth they provide. However, they are still, more or less, on their early stage of development and the need for data compression will still remain even with these technologies.

Chapter 7

SYSTEM ARCHITECTURE

In the previous chapters we have examined various methods for improving the software implementation of a fractal video encoder and decoder, as well as the basic concepts of networking and data distribution. In this chapter we consider the real–time implementation of these algorithms and their integration into a video–conferencing system. We will therefore describe the different hardware sub–systems that are likely to be involved on such a fractal video conferencing system. In addition the X window interface will be introduced and several X applications made for experimental purposes will be presented. Finally we will describe the architecture of the system.

7.1. State–of–the–art of distributed video systems

Before we proceed to describe the hardware and software sub–systems of video–conferencing systems it is necessary to briefly review the state of the art of these applications. Many desktop distributed video systems have been developed in academia and industry and more are under investigation. Most of these systems perform relatively well although problems of speed and image quality do exist.

In particular the industrial products provide many features such as TCP/IP multi–cast capability for multiple broadcasting to many viewers over modem, ISDN, Internet and Ethernet connections. They also support video e–mail, video answering machine and other recording options and collaboration tools for conferencing which make the products attractive solutions to the market. Most of these systems are based on the H.261 video compression standard which is usually implemented on special video compression hardware boards with application specific VLSI circuitry so that it won't burden the host at all.

In academia researchers are concentrating in the provision of audio and video signal across the internet by developing advanced protocols for data handling. This is because the research community has a different environment and requirements which usually can not be fulfilled by the commercial services. The most common compression technique used for the video signal is the H.261 standard. It is only recently that the MPEG standard [61] has become more popular for such projects and its use is being investigated. We will briefly present some of these systems starting from the Pandora system.

The Pandora system [60] was one of the early efforts to produce a network of multimedia workstations able to manipulate and distribute in real–time video and audio signals. It was designed as a local area network and was based on a central box (the pandora box) which consisted of a video capture card, the compression hardware and software, the server and the mixer sub–systems. The compression algorithm that was used exploited the property of images that adjacent pixels very often have similar values and instead of sending the actual values of the pixels the difference from the previous pixel was sent.

The Multimedia Integrated Conferencing for European Researchers (MICE) and the MBone systems [62] are more recent efforts which aim to provide multi–media inte-

grated conferencing service between different European sites. The H.261 standard was used for the compression of the video in both cases.

Finally distributed, real–time, MPEG video and audio players across the Internet have recently appeared providing effective mechanisms for client/server synchronization and good performance. In some cases distributed movie systems support many different compression techniques (MPEG, H.261, Motion JPEG) simultaneously.

We can see from the above that there is currently significant activity in the area of distributed video and sound for wide and local area networks. Many different approaches and products compose the total effort for the provision of reliable and high–quality real–time distributed multi–media. Although most of these efforts are based on the established video compression standards, the philosophy and architecture of these system is similar to the fractal video–conferencing system that we propose. It is reasonable therefore that our system will include similar hardware components and use a similar software interface.

7.2. Image processing hardware

Electronic imaging has been dependent to a great extent upon sub–systems specifically developed for processing large amounts of data. These sub–systems typically have been board–oriented to address the tasks of grabbing, processing, storage, displaying and in some instances transmitting the high data content of all varieties of real world images (both grayscale and colour) in a timely and cost effective manner. Such a board–oriented image processing system, produced by Imaging Technology, is used for our project. It is based on the VME–bus and consists of three cards:

- The ADI–150 *analogue to digital converter.* This is a device that converts the electrical output of the physical sensing device (such as a camera) into digital form. It has four separate camera inputs which are software selectable. A mono-

chrome video signal can be digitized in real time and routed to the frame buffer for interim storing.

- The FB–150 *frame buffer.* This is used for the specialised digital storage that is usually required by the image processing system. In order to meet these needs the frame buffer has to be able to store one or more entire digital images and be accessed at the digitisation rate of the analogue to digital converter (normally 30 complete images per second). The amount of storage in the frame buffer is limited by the physical size of the card and by the storage density of the memory chips used. In our case the frame buffer contains two sixteen–bit frame stores which give the capability to store four different images simultaneously.

- The ALU–150 *arithmetic logic unit.* This is the basic processing unit whose function is to perform arithmetic and logic operations in parallel, typically at video frame rates (for European standard video this means an arithmetic or logic operation between two images in 1/25 sec). In order to be able to process the large amount of video data in real–time a high bandwidth connection is required between the the frame buffer and the ALU. Usually the simple image processing operations such as thresholding, convolution and image addition/subtraction that are performed on it are considered to be limited compared with the complexity of some novel image processing algorithms. Thus complex algorithms can be more efficiently implemented on a general–purpose computer.

The above system which is known as the *Itex* system, has been interfaced to a Personal Computer (PC), based on the Intel's 80286 processor which is running the Linux operating system. The interfacing was made using a PC–bus to VME converter card. In our project the above hardware will only be used as a frame grabber following the steps below:

- The system initially acquires real world, gray scale, images via monochrome CCD[3] digital cameras which are used as the acquisition medium. These cameras are connected to the camera input ports of the itex system;
- The images are then digitised in the analogue to digital converter;
- Finally they are stored in the frame buffer and wait to be processed.

The arithmetic and logic unit will not be considered at all since its signal processing ability is limited compared to the complexity of the fractal compression algorithm. For the implementation of this algorithm special programmable digital signal processing hardware has to be used.

7.3. The TMS320C40 Parallel Digital Signal Processor.

In the last few years the performance of processors has increased dramatically, allowing the implementation of an increasing number of applications particularly in the areas of imaging, graphics and multi–media, where time consuming data processing algorithms are needed. However in many cases the need for even faster processors is pressing and a situation in which a single processor is not able to serve the application needs becomes more and more common especially in the area of digital signal processing.

As a solution to the problem, the use of multiple processors to execute a single task is adopted by many researchers. This can be achieved in two ways:

- The first one is to partition the application into several tasks and for each one of the processors to work independently and simultaneously on different tasks or groups of tasks.

3. Charge–Coupled Detector Camera

- The second deals with the concurrent execution of portions of a single application.

Since the fractal video compression algorithm examined in chapter 5 is still far from a real – time implementation on a Sun Sparc IPC workstation we will investigate the use of a parallel DSP in order to speed it up. The Texas Instruments TMS320C40 floating–point processor is considered for this, since it offers a parallel architecture with high performance. The processor can be programmed in a high level language (C) and it is provided with libraries supporting multiple processor systems. The latter feature enables the easy extension of the system with added processors for more computational power which will guarantee video coding in real–time[58].

In more detail, the TMS320C40 processor provides both on–chip and off–chip parallelism. The on–chip parallelism relies on architectural enhancements for improved performance. The TMS320C40 uses a superscaling architecture which means that instead of breaking the instruction pipeline into its distinct stages as in the superpipelining architecture or using multiple CPUs as in the multi–CPU integration, it includes multiple pipelines within a processor allowing the CPU to execute multiple instructions simultaneously[58].

The off–chip parallelism incorporates the linking of several processors together via parallel or serial communication ports (links). TMS320C40 uses six parallel ports to create topologies (theoretically of any size) consisting of many processors in order to meet the needs of any application. In addition the TMS320C40 has an on–chip, six–channel DMA[4] coprocessor which supports very fast (20Mbyte/s) transfers at each communication port [59]. The DMA can be used to handle I/O operations from external memory and peripherals, allowing in such way the CPU to focus its entire performance on other computational tasks.

4. Direct Memory Access

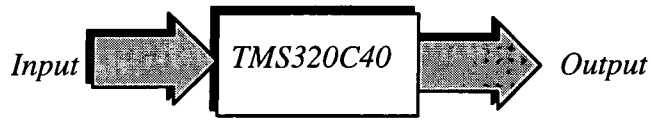


Figure 7.1: *Von Neuman (SISD) computer with C40.*

In order to roughly evaluate the TMS320C40 potential as a node for a multi–processor system suitable for video applications using fractal compression, the algorithms were tested on a single C40 processor board. This board was interfaced to an 80486 based PC which was running DOS. As the development software for the TMS320C40 board, including the drivers for the interfacing and the C compiler, runs on DOS systems, the frame grabber, whose drivers were written for a different operating system, had to be connected to a different PC.

In these experiments the TMS320C40 was used in a Single–Instruction / Single–Data stream (SISD) topology (Von Neuman computer), as illustrated in figure 7.1. Although the C compiler allows the implementation of parallel programs using threads and tasks, we decided to implement the algorithms in a sequential manner to avoid the communication overheads between parallel processes, since there was only a single processor available.

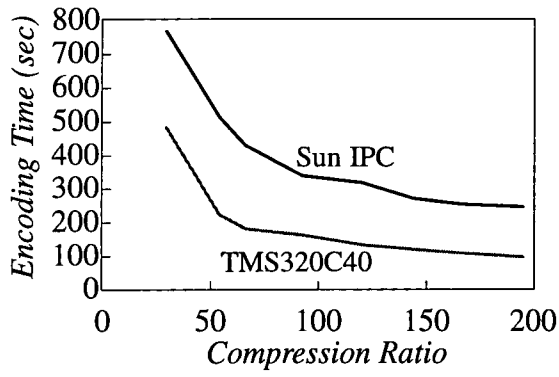


Figure 7.2: *Comparison of the time needed to encode 100 frames of the 'Miss America' sequence on the SUN IPC station and on the TMS320C40.*

Results of these experiments can be seen in figure 7.2. In this diagram the time needed to execute 100 frames of the 'Miss America' sequence in the TMS320C40, for different compression ratios, are illustrated and compared to the corresponding results in the Sun IPC SPARC station as described in chapter 5. We can see that the algorithm runs considerably faster on the TMS320C40 hardware. The average decrease in the execution time is more than 2 times, allowing the encoding of a frame in less than one second (1.02 frames per second) in the best case. The decrease on the execution time is expected to be even larger if the algorithm optimisation was improved. In particular, some of the routines which play a major role in the efficiency of the algorithm (i.e. for matrix multiplication), could be coded in the TMS320C40 low–level assembly language rather than the standard parallel C avoiding the inefficient code produced by the C compiler.

7.4. The X Windows Protocol

The X Windows system is a windowing graphical environment which allows the execution of multiple applications in different rectangular windows. It provides facilities to generate text and two–dimensional monochrome or colour graphics on bit-mapped displays. In these displays each pixel corresponds to one or more bits in memory and is controlled independently.

The above functionality can be found in other windowing systems. However it was the X Windows environment which became a standard because of a special functionality that it offers: the network transparency. This means that users are able to run application programs on remote machines throughout the network, as if they were on their local machines. A direct consequence of this is that X Windows applications can be implemented on different computer architectures and operating systems in a device independent way allowing cooperation between a network of different computers. For

example, a computationally intensive application may take input from a workstation, run on a super–computer and return the results across the network to be printed or plotted back on the workstation's display.

In order to provide this functionality, the X Windows system was designed as a network protocol, based on the client/server concept. In this case the program that controls each display is considered as the *server* since this makes the displays accessible to other systems across the networks, passes user input to the clients and is responsible for the drawing of the graphics and text. *Clients,* on the other hand, are considered to be the application programs that run using the network services of the window system. Clients can connect to any display on which they are allowed to.

One such client with special authority is the window manager. This program is responsible for the screen layout and appearance. It allows the user to manipulate client windows by moving, resizing or iconifying them and to start new applications.

In practise, for the creation of new X applications, it is common to utilize some object–oriented toolkits. These toolkits allow the programmer to build the interface of his application, using configurable user interface components such as buttons, sliders and pull–down menus, which are known as widgets. There are several X Windows toolkits available, including XView and Motif. The XView toolkit was chosen for this work, since it is distributed in public domain, and provides a good functionality.

The design of the toolkit encourages event–driven programming [64]. This means that the program remains inactive unless the user generates an event by pressing for instance a button or moving the mouse cursor. In this way the user controls more directly the execution flow.

While experimenting with the XView toolkit some examples of X applications, were developed. These are shown in figures 7.3 and 7.4. The first application implements graphically the decoding of a simple IFS code after a certain number of iterations using

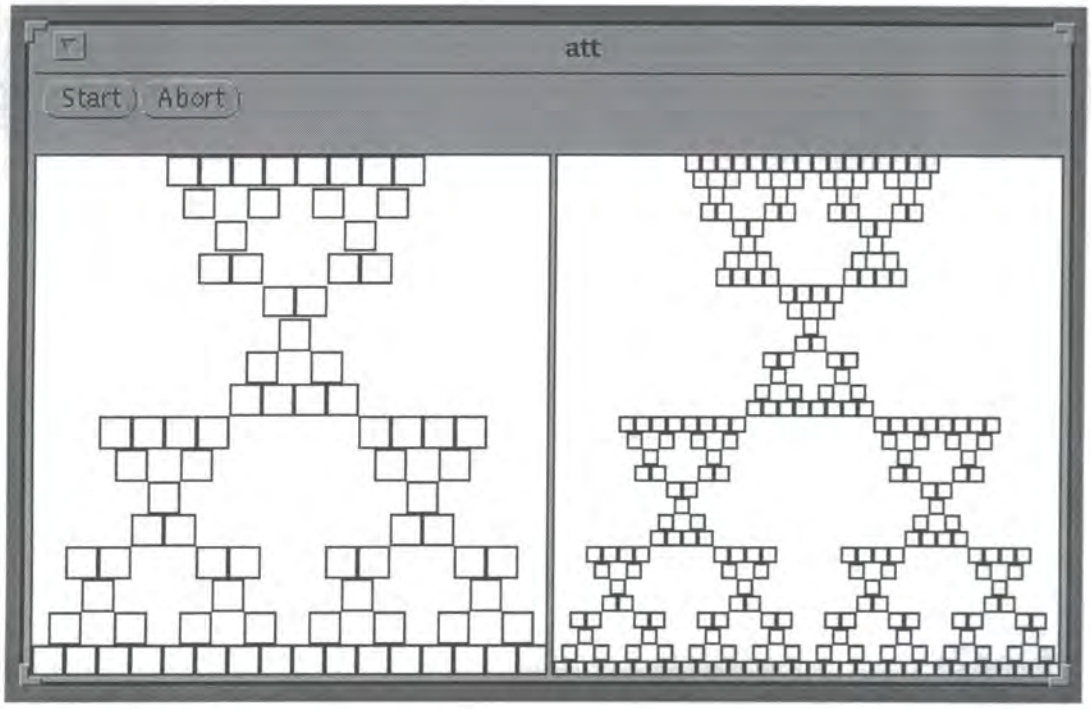
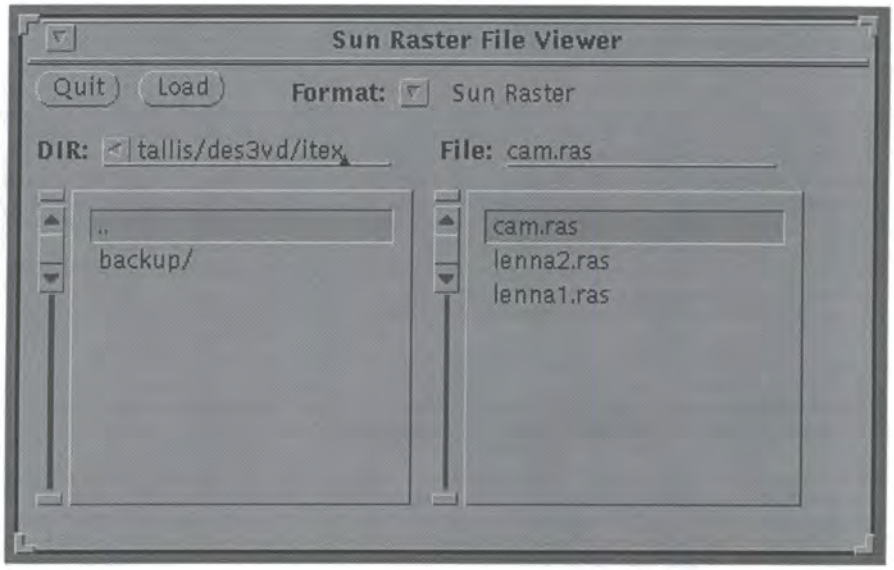


Figure 7.3: *X Windows application which graphical represents on two canvases, the iterative decoding process of an IFS using the photocopy machine algorithm.*

the photocopier machine algorithm. The two canvases on the single frame were used to illustrate in every case two consecutive iterations of the same IFS and therefore understand in a better way the convergence of the IFS on the same attractor after a few iterations.

The second figure shows a sun raster (bitmap format) image viewer which was created as a first step for the XVideo application which follows. This viewer consists of two windows. In figure 7.4. (a) the first window (*parent*) is illustrated. This window recognizes and presents on the left and right menus all the subdirectories below the current directory, as well as all the image files of sun raster format in the current directory. If the user wants to move one directory up, the two dots character ('..') at the beginning of the first menu have to be pressed. By choosing an image file and pressing the 'Load' button a sub–window (child window) like the one illustrated in figure 7.4 (b)



(a) Parent Window

(b) Child Window

Figure 7.4: *X Viewer for Sun Raster format images.*

will appear which shows the image in a canvas. There are two important points regarding both applications that need to be discussed in more detail.

The first one is that for the manipulation of the pixels, calls to special Xlib [63] procedures are used. The Xlib is a library which contains easy to use routines for drawing points, lines, rectangles, circles, strings and other objects as well as combinations of objects. These routines need only a few arguments which denote the object's size and coordinates as well as the Graphics Context (GC) which is the essential

controller of the appearance of the object within a window. The use of the library is essential to ensure portability of code between XWindows systems.

The second point concerns, the *colourmap* of the image files which is in fact a colour lookup table. When we are dealing with black and white displays, each pixel can only have two states and therefore it is represented by a single bit. The value of that bit defines the state of the pixel. If we are dealing, with monochrome (gray scale) or colour systems, then each pixel is represented by multiple bits which do not directly control the pixel value (colour) but instead they define an index to the colourmap as illustrated in figure 7.5.

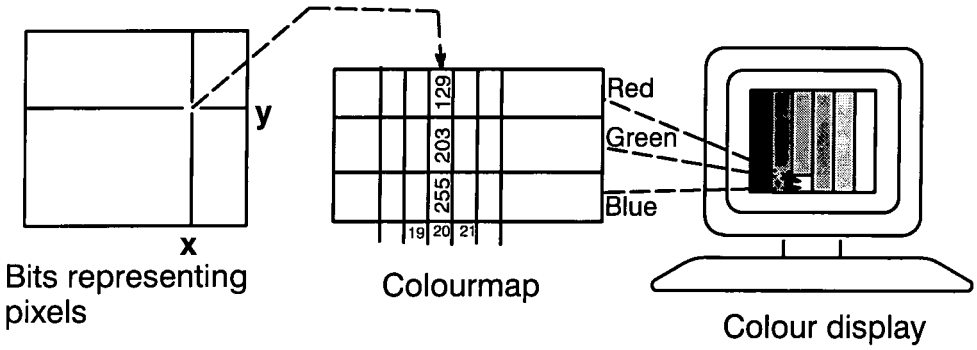


Figure 7.5: *Mapping of Pixel value to Colour using colourmaps.*

On a colour display this is an array of Red, Green and Blue (RGB) intensity values since the combination of these three colours is enough to create all the known colours. For instance, suppose that the pixel at x,y position is assigned by a bit set of value 20. Then the RGB values of the colourmap that correspond on this cell (called *colourcell*) are displayed on the screen at that location. All bitmapped displays have at least one hardware colourmap, although in the case of a single–plane monochrome screen it may consist of only two colourcells.

In general, there is a very large number of possible RGB combinations; a typical display might have 8 bits assigned to every pixel, while its colourmap entries are 24 bits long, 8 bits to describe each of the red, green and blue intensities. This means that a

total of more than 16 million (256^3) colours are supported. In practice it is often not possible to display more than 256 colours simultaneously and so a small subset of the colourmap will be displayed at a time.

The use of a colourmap may present a significant problem when two clients share the same display screen since they must use the same hardware colourmap. For example, suppose there is an application running on the display, that requires the use of a gray scale monochrome colourmap. If at the same time and on the same display another application requires green and blue colours then strange artifacts and wrong colour allocations may occur resulting at images such as the one shown in figure 7.4.(b). In practice, the different applications use certain mechanisms to communicate and decide to allocate a common entry in the hardware colourmap which is visually close to their requirements. Alternatively, the use of *private* colourmaps is possible. In this case the colourmap of the application is swapped in only when the application window is selected by the mouse pointer. In this case the other applications on the screen would appear in unusual colours. For our applications the first solution was adopted.

Initial experimentation with the XWindows environment indicated that the implementation of an XVideo application should be possible. In our XVideo application fractally encoded frame sequences were loaded one by one, then decoded and finally displayed as fast as possible on the same canvas in order to give the illusion of motion on the object. The window of this application is shown in figure 7.6. The current frame number is displayed between the two buttons 'Start' and 'Quit'. It can be seen that while the program is running the 'Stop' button becomes gray which means that it is inactive.

Figure 7.6: *X Video: It decodes fractally encoded frames and displays them as fast as possible on the canvas.*

7.5. System Implementation and Discussion

Considering all the individual software and hardware components that were discussed above we can propose that a hypothetical fractal video conferencing system, which uses these components would look like the one shown in figure 7.7. A CCD camera would be connected in the input ports of a frame grabber, such as the one described, which converts the analogue signal into digital and stores it into a buffer at normal video frame rates.

The frame grabber system would be interfaced on a PC running Linux since the latter operating system provides the TCP/IP protocol and the libraries needed to create sockets, which allows the easy interconnection of the PC to a local or wide area network. Alternatively a UNIX workstation with an attached frame grabbing device can be used instead of the PC, since this provides higher bandwidth data bus which would allow the faster transfer of image data into the server computer memory.

In order to speed up the encoding process, a parallel topology consisting of an adequate number of parallel processors, such as the TMS320C40 DSP, could be used.

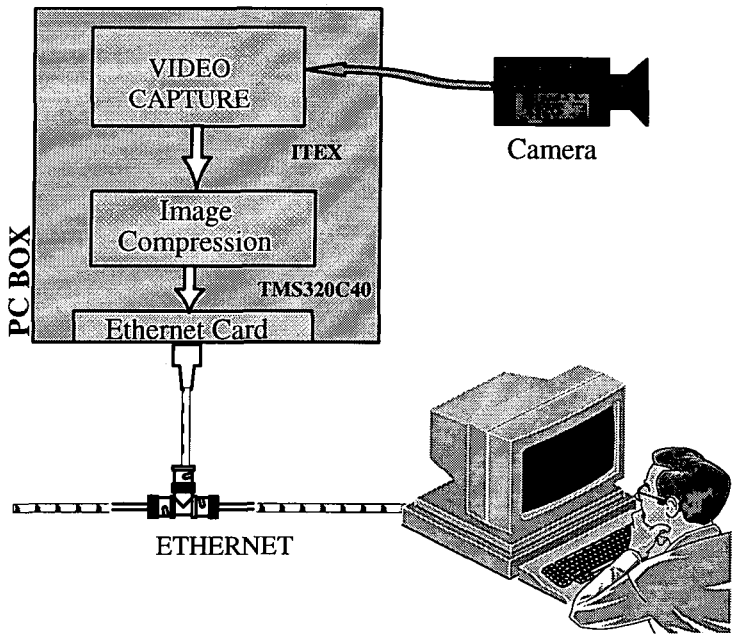


***Figure 7.7**: Architecture of the Fractal Video Distribution system*

The number of processors needed for a real–time operation, depends on the parallel implementation of the algorithm on a suitable topology. Therefore, the exact number of processors can only be estimated through experimentation. However, if we consider in a very simplistic way the results obtained from the execution of the fractal compression algorithm on the single processor, we can estimate that around 30 processors are needed for a real–time implementation, since interprocess communication latency exists. Although this is a relatively large number of processors which clearly increases the cost of such system, it is not completely unrealistic, as high performance computers are becoming more and more common and available at low cost.

The parallel hardware topology has to be attached to the same PC or workstation box and be able to communicate with the buffer of the frame grabber in order to transfer the stored images. In both cases the appropriate drivers have to be designed to enable the interfacing of the new hardware. For best input/output results the built–in DMA co–processor of the TMS320C40 processor can be used. In this case the frames are transferred from and into memory using different channels, without interfering with the

operation of the CPU. Therefore interfacing to slow external memories and peripherals is allowed without reducing the performance of the CPU.

Since standard Ethernet is a common network medium for TCP/IP connection, we will consider that the remote machines are interconnected on the same Ethernet network. The Ethernet network has a 10MBit/sec bandwidth. For the interconnection of the PC to such a network, an Ethernet card is needed in the PC. A server would be developed and run on this PC which would be able to store and replay encoded frames while listening for connection requests by a client in remote hosts such as the two Sun Sparc stations on which our experiments have been conducted. These workstations run SunOS 4.1.2 and are also interconnected on the 10 MBit/sec Ethernet network.

The system will then operate as follows: The ITEX system captures frames which are sent to the DSP for compression, via the DSPs DMA I/O coprocessor's channels. The image frames are then sent back to the PC and stored in memory. Simultaneously the PC waits for connection requests through the network. Normally more than one image frames is stored in memory before transmission. When such a client request is received connection between the PC and the remote host is established and the last fully downloaded and compressed frame is transferred via the network to the workstation using the TCP/IP or UDP/IP protocols.

In the X window environment, a window showing the decompressed images that are received is produced. The X client periodically connects to the PC and receives the frame data into memory. The data is then decompressed and converted into X image format before being sent to the display server.

If we consider gray–scale frames of 256x256 size, these would need a bandwidth of more than 13MBit/sec to be transferred in real–time (25 frames per second) while the bandwidth of the current Ethernet network is an absolute maximum of only 10MBit/sec. In the current implementation of the fractal video compression algorithm, the compres-

sion ratios that can be achieved while the image quality remains acceptable, can be more than 150 and a typical value for the compression ratio is more than 60. The latter means that the bandwidth needed to transfer in real–time the compressed images ranges from 87.4KBit/sec up to 200KBit/sec. In other words by applying the fractal video compression technique only an 1.5% of the available bandwidth is used in average. Therefore, a large bandwidth remains, which could be used to transfer audio files or other information.

7.6. Summary

In this chapter we described the hardware and software components that are needed for the implementation of a fractal video conferencing system and we presented experiments which investigated the capabilities of them. The first component was the image processing hardware which includes the camera, an A/D converter and a buffer. This component serves as a frame grabbing device in the system.

The second one was the TMS320C40 parallel DSP which could be used to speed–up the encoding process of the frames as part of a parallel topology. Although results showed a significant increase in the performance of the fractal compression it can be concluded that a parallel topology consisting of a relatively large number of processors would be needed for implementing the algorithm in real–time.

The third component, the X protocol, was also introduced as the graphical interface of the video conferencing system. Several applications which were developed while experimenting with it are presented. The final application, the XVideo, was implementing a fractal video decompressor and viewer on the X display.

Finally a hypothetical fractal video conferencing system based on the above components and other existing hardware such as workstations, PCs and Ethernet network was described. One of the major conclusions that can be extracted is that a fractal video

conferencing system can be implemented on a local area Ethernet network. The encoded video frames can be transmitted in real–time since they are compressed at very low compression ratios and only occupy a very small part (1%–2%) of the total bandwidth. However ~~the use of~~ a large number of processors has to be used to implement the fractal compression technique in real–time and clearly cost will increase in this case.

Chapter 8

CONCLUDING REMARKS

In this thesis, we have investigated the use of fractal compression technique for a real–time video distribution system through telecommunication networks (currently Ethernet). The current activity in the field of image and video compression has been described. Much of this work aims to provide compression techniques which combine large compression ratios, high quality reconstructed images and real–time implementations. We have shown that most of the state–of–the–art compression techniques do not efficiently combine all of these aspects of image compression.

The fractal compression technique has been considered as an alternative to existing standards. Fractal block coding has some useful properties which satisfy many of the requirements for video streams. These properties include the scaling of image to arbitrary sizes, large compression ratios and fast, hierarchical decompression. The basic algorithm upon which all the fractal compression schemes are based was initially described. We have shown that the fractal technique performs quite well for arbitrary gray scale images but suffers from long execution times. Several schemes for improving the algorithm have been proposed and examined for both still images and image sequences.

Most of the fractal schemes examined in this thesis aimed at the reduction of execution time, in order to make the fractal compression method suitable for use in a video–conferencing system. Amongst these, two novel concepts in the context of still fractal image compression have been presented. In the first, the image was partitioned into sections and each section was encoded independently. The results obtained were very encouraging, but the compression time remained relatively long in the final implementation. Another effort that produced unsatisfactory results was the use of the median metric as an alternative to the RMS. The RMS metric was finally preferred over the median metric as a more efficient implementation. In addition, the hierarchical reconstruction of the encoded image was presented in order to reduce the decompression time.

The extension of the improved fractal compression scheme to video sequences was then considered. Three schemes were examined which all involved the compression of frames with respect to their predecessor, in order to reduce the temporal redundancy. In the final and most efficient implementation, significant improvement in performance was obtained but, even in this case, real–time execution was not achieved. In particular the compression process has to be speeded up almost 70 times to achieve real–time execution while the decompression process has to be speeded up by a factor of 7. The latter should be achievable on more powerful workstations than those used in this work.

We can conclude therefore, that in the current implementation, the fractal video compression technique can not compete with the state–of–the–art video coding algorithms (i.e. MPEG or H261 standards). Its major drawback, which is that the execution of the compression process is time consuming, still remains although not to the same extent as in the early implementations of the algorithm.

Data distribution and networks were also introduced as an essential element of a video distribution system. The use of the TCP/IP and UDP/IP protocol architectures

were considered as they are the most widely used. These are generally used on UNIX machines and they provide an efficient way to create new applications using sockets or RPCs.

Other hardware and software components of a video–conferencing system were also examined separately and experiments described in this thesis demonstrate that these components can be implemented in an XWindows environment. These components were the frame grabber hardware, the XWindows protocol and a single TMS320C40 parallel processor. The latter was considered in order to evaluate the speed improvements of the compression algorithm on a powerful hardware system which may consist of many parallel processors. Although the use of the DSP processor significantly reduced the compression time, a relatively large number of processors would be needed to implement the current algorithm in real–time.

The experiments reported in this thesis indicate that a compression speed of 1 frame per second can be achieved using a single TMS320C40 processor. A very simplistic extrapolation of this result suggests that around 30 such processors could achieve real–time compression of a video signal. This estimate does not take into account communication overheads in such a parallel system. However, this is not considered to be too significant in such a computationally intensive process. Furthermore, more sophisticated parallel architectures and the use of image partitioning as described, could reduce the number of processors below this estimate. Clearly, a system of 30 TMS320C40 processors is not commercially viable but further development of dedicated hardware may improve this situation in the near future.

Generally speaking, the fractal block coding method has great potential. It is a recently developed technique with many advantages. The only major disadvantage is the relatively long execution times. However, there is a promise for improved performance in the near future given further investigation and research into the technique. High

performance computer systems are becoming more common as very powerful processors become available at lower cost. Therefore, if further consideration is given to optimisation of the software code, and other schemes for improving the efficiency of the fractal compression methods are investigated, real–time execution of the compression and decompression process using fractal compression techniques may be achieved in the near future.

Bibliography

[1] C.Brown, *UNIX–Distributed Programming,* Prentice Hall International (UK), 1994.

[2] M.Sloman and J.Kramer, *Distributed Systems and Computer Networks,* Prentice Hall International (UK), 1987.

[3] M.Barnsley and D.A.Sloan, *A Better Way to Compress Images,* BYTE, pp.215–223, 1988.

[4] POEM ColorBox, *User's Guide Version 2.0,* Iterated Systems, Inc. Norcross Georgia, 1993.

[5] M.Barnsley, *Fractals Everywhere,* Academic Press, 1988.

[6] A.E.Jacquin, *Image Coding Based on a Fractal Theory of Iterative Contractive Image Transformations,* IEEE Trans. on Image Processing, Vol.1, No.1, pp.18–30, 1992.

[7] G.Wallace, *The JPEG Still Picture Compression Standard,* Communications of the ACM, Vol.34, No.4, pp.30–44, 1991.

[8] C.Frigaard, J.Gade, T.Hemmingsen and T.Sand, *Image Compression Based on a Fractal Theory,* Internal Report, Institute of Electronic Systems, Aalborg University, Denmark, pp.1–10, 1994.

[9] M.L.Hilton, B.D.Jawerth and A.Sengupta, *Compressing Still and Moving Images with Wavelets,* Multimedia Systems, Vol.2, No.3, 1994.

[10] B.Liu and A.Zaccarin, *New fast algorithms for the estimation of block motion vectors,* IEEE Trans. Circuits Systems for Video Technology, 3(2):148–157, 1993.

[11] A.S.Lewis and G.Knowles, *Video compression using 3D wavelet transforms*, Electronics Letters, 26(6):396–397.

[12] T.R.Reed, *High quality image compression using the Gabor transform,* Proceedings of SID'93, pp. 792–795, Seattle.

[13] JPEG–FAQ: *Frequently Asked Questiones for JPEG image compression*, FTP access from ftp://rtfm.mit.edu/pub/usenet/news–answers/jpeg–faq.

[14] D.LeGall, *MPEG: A Video Compression Standard for Multimedia Applications,* Communications of the ACM, Vol.34, No.4, pp.46–58, 1991.

[15] MPEG–FAQ: *Frequently Asked Questiones for MPEG image compression*, Available on the WWW at: http://www.crs4.it/HTML/LUIGI/MPEG/mpegfaq.html.

[16] R.M.Gray, *Vector Quantization,* IEEE ASSP Magazine, Vol.1, No.2, pp.4–29, 1984.

[17] W.Zettler, J.Huffman, D.C.P.Linden, *Application of Compactly Supported Wavelets to Image Compression,* Image Processing Algorithms and Techniques, Proc. SPIE, Vol.1244, pp. 150–160, 1990.

[18] N. Sonehara, *Image Data Compression Using a Neural Networks Model,* Proc. International Joint Conference on Neural Networks (IJCNN'89). IEEE, 1989.

[19] E.Gelenbe and M.Sungur, *Image Compression with the random neural network,* Proc. International Conference on Artificial Neural Networks, North–Holland Elsevier, 1994.

[20] E.T.Copson, *Metric Spaces,* Cambridge University Press, 1968.

[21] Y.Fisher, *Fractal Image Compression: Theory and Applications*, Springer–Verlag, 1995.

[22] M.Barnsley and L.P.Hurd, *Fractal Image Compression*, AK Peters, Ltd., 1993.

[23] Z.Baharav, D.Malah and E.Karin, *Hierarchical Interpretetion of Fractal Image Coding and Its Application to Fast Decoding,* International Conference on Digital Signal Processing, Cyprus, July 1993.

[24] F.Dudbrodge, *Fast Image Coding by a Hierarchical Fractal Construction,* submitted, pp. 1–6, 1994.
Found on: [ftp://links.uwaterloo.ca:/pub/Fractals/Papers/External/dudbridge95b.ps.gz]

[25] Z.Baharav, D.Malah and E.Karnin, *Hierarchical Interpretation of Fractal Image Coding and Its Applications,* in Fractal Image Compression: Theory and Application to Digital Images (Y.Fisher, Ed.), New York: Springer–Verlag, 1995, pp.231–244, 1995.

[26] Y.Fisher, *Fractal Image Compression,* SIGGRAPH '92 Course Notes, vol.12, pp. 7.1–7.19.
[ftp://links.uwaterloo.ca:/pub/Fractals/Papers/External/fisher92a.ps.gz]

[27] M.Lazar, L.Bruton, *Fractal Block Coding of Digital Video,* IEEE Transactions on Circuits and Systems for Video Technology, vol.4, no.3, pp.297–308, 1994.

[28] E.Reusens, *Sequence Coding Based on the Fractal Theory of Iterated Transformations Systems,* Proceedings of the SPIE: Visual Communications and Image Processing, vol.2094, pp.132–140,1993.

[29] W.O.Cochran, J.C.Hart, P.J.Flynn, *Principal Component Classification for Fractal Volume Compression,* Proceedings of Western Computer Graphics Symposium, pp. 9–18, Mar. 1995.

[30] A.Bogdan, *Multiscale (Inter / Intra–Frame) Fractal Video Coding,* IEEE International Conference on Image Processing ICIP '94, 1994.

[31] B.Bani–Eqbal, *Speeding up Fractal Image Compression,* Proceedings of SPIE: Still Image Compression, vol.2418, pp.67–74, 1995.

[32] K.U.Barthel, T.Voye and P.Noll, *Improved Fractal Image Coding,* PCS '93 – Proceedings of the International Picture Coding Symposium, section 1.5, 1993.

[33] Y.Fisher, T.Shen and D.Rogovin, *Fractal (Self–VQ) Video Encoding,* Proceedings of the SPIE: Visual Communications and Image Processing '94, vol. 2304, 1994.

[34] Y.Fisher, T.Shen and D.Rogovin, *A Comparison of Fractal Methods with DCT and Wavelets,* Proceedings of the SPIE: Neural and Stochastic Methods in Image and Signal Processing III, vol.2304–16,1994.

[35] C.Frigaard, *Fast Fractal 2D / 3D Image Compression,* Internal Report, Institute for Electronic Systems, Aalborg University, Denmark, pp.1–81,1995.
[ftp://links.uwaterloo.ca:/pub/Fractals/Papers/External/Frigaard95.ps.gz]

[36] B.Hurtgen, P.Buttgen, *Fractal Approach to Low Rate Video Coding,* Proceedings of the SPIE: Visual Communications and Image Processing, vol.2094, pp. 120–131, 1993.
[ftp://links.uwaterloo.ca:/pub/Fractals/Papers/External/hurtgen93c.ps.gz]

[37] J.Kominek, *Algorithm for Fast Fractal Image Compression,* Proceedings of SPIE: Digital Video Compression: Algorithms and Technologies 1995, vol. 2419, pp. 296–305, 1995.
[ftp://links.uwaterloo.ca:/pub/Fractals/Papers/Waterloo/kominek95a.ps.gz]

[38] J.Kominek, *Advances in Fractal Compression for Multimedia Applications,* Multimedia Systems Journal, submitted, 1995.
[ftp://links.uwaterloo.ca:/pub/Fractals/Papers/Waterloo/kominek95c.ps.gz]

[39] D.Monro and J.Nicholls, *Real Time Fractal Video for Personal Communications,* Fractals – An Interdisciplinary Journal on the Complex Geometry of Nature, vol.2, no.3, pp. 391–394, 1994.

[40] D.Saupe, R.Hamzaoui, *A Guided Tour of the Fractal Compression Literature,* ACM SIGGRAPH'94 Course Notes – Course 13: New Directions for Fractal Modelling in Computer Graphics, 1994.
[ftp://links.uwaterloo.ca:/pub/Fractals/Papers/External/saupe94b.ps.gz]

[41] D.Saupe, *Accelerating Fractal Image Compression by Multi–Dimensional Nearest Neighbor Search,* IEEE Data Compression Conference Proceedings '95, James Storer (editor), pp. 222–231, 1995.
[ftp://links.uwaterloo.ca:/pub/Fractals/Papers/External/saupe95b.ps.gz]

[42] D.Wilson, J.Nicholls and D.Monro, *Rate Buffered Fractal Video,* submitted to Proceedings of ICASSP, 1994.

[43] D.J.Jackson and T.Blom, *Fractal Image Compression Using a Circulating Pipeline Computation Model,* submitted in the International Journal of Parallel Programming, 1995.
[http://www.carl.ua.edu/people/jjackson/home.html]

[44] D.J.Jackson and G.S.Tinney, *Performance Analysis of Distributed Implementations of a Fractal Image Compression Algorithm,* to be published in Concurrency: Practice and Experience, 1995.
[http://www.carl.ua.edu/people/jjackson/home.html]

[45] D.Monro and S.Wooley, *Fractal Image Compression Without Searching,* Proceedings of ICASSP, 1994.

[46] J.McConnell, *Internetworking Computer Systems,* Prentice–Hall, 1988.

[47] J.R.Freer, *Computer Communications and Networks,* Pitman, London, UK, 1988.

[48] P.Hardy, *Introducing Data Communication Protocols.* NCC Publications, Manchester, UK, 1985.

[49] F. Halsall, *Data Communications, Computer Networks and OSI,* Addison–Wesley Publishing Company, Second Edition. 1988.

[50] V.C.Marney–Petix, *Networking and Data Communication,* Reston Publishing Company Inc, A Prentice–Hall Company, 1986.

[51] International Organization for Standardization, *Open System Interconnection: Basic Reference Model, ISO 7498,* Geneva, Switzerland, 1984.

[52] G.V.Bochmann, *Protocol Specification for OSI,* Computer Networks and ISDN Systems, vol. 18, pp.167–184, 1990.

[53] J.Fritz, *You can take it with you,* Byte magazine, September 1995.

[54] L.Tropiano and D.Mcnutt, *How to implement ISDN,* Byte magazine April 1995.

[55] R.Barnett and S.Maynard–Smith, *Packet Switched Networks: Theory and Practice,* Sigma Press, 1988.

[56] D.C.Lynch and M.T.Rose, *Internet System : Handbook,* Addison–Wesley Publishing Company, Inc, 1993.

[57] M.J.Jubb and A.Purvis, *Analysis of Network Protocol Performance in the Context of Multi–Workstation Parallel Distributed Applications,* Microprocessing and Microprogramming, Vol.40, No.10–12, pp.807–810, 1994.

[58] Texas Instruments, *TMS320C4x User's Guide, 1991.*

[59] Texas Instruments, *Parallel Processing with the TMS320C4x : Application Guide,* 1994.

[60] A. Hopper, *Pandora – an experimental system for multimedia applications,* ACM Operating Systems Review 24, 2 1990.

[61] K.Patel, B.C.Smith and L.A.Rowe, *Performance of a software MPEG Video Coder,* Proceedings of first ACM International Conference on Multimedia (Anaheim, CA, Aug. 1993), pp.75–82.

[62] S.Casner, *Are you on the MBone?*, IEEE Mutlimedia 1, 2, pp.76–79, 1994.

[63] A.Nye, *Xlib Programming Manual for version 11 of the X Window System, O'Reilly & Associates, Inc. 1988.*

[64] D.Heller, *XView Programming Manual for XView version 3,* O'Reilly & Associates, Inc. 1991.