FRACTAL BLOCK CODING TECHNIQUES IN IMAGE COMPRESSION

BY

JINSONG LIU M.Sc

This thesis is submitted as the fulfilment of the requirement for the award of the Degree of PhD by research to:

DUBLIN CITY UNIVERSITY

Supervisor: Dr. Sean Marlow

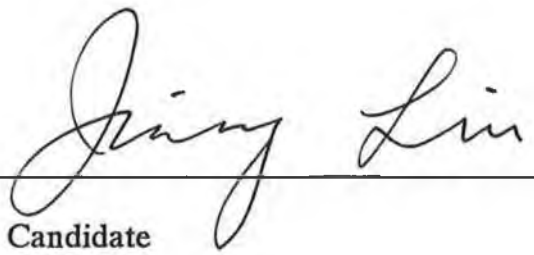
School of Electronic Engineering

Dublin City University, IRELAND

November 1994

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of PhD is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:  I.D. No: 90300777

Candidate

Date: November 1994.

ACKNOWLEDGMENTS

The author wishes to express his sincere gratitude to Dr. Sean Marlow, Dr Noel Murphy and Dr. Michael Scott for acting as supervisors and for their kind guidance, helpful comments, valuable assistance and encouragement at all stages of this research.

The author is indebted to the Telecom PAT of School of Electronic Enginerring in Dublin City University for providing financial support during the course of this research.

Thanks are also due to my fellow research students, Liam Ward, Noel Brady and Noel O'Connor of video coding group of the School of Electronic Engineering, for their kind help towards my research work.

CONTENTS

ABSTRACT

Fractal block coding is a relatively new scheme for image compression. In this dissertation, several advanced schemes are proposed based upon Jacquin's fractal block coding scheme. Exploiting self-similarity at different target block size levels is proposed which allows the self-similarity in the image to be exploited further. Smoother areas are coded with bigger target block sizes while fine details are coded with smaller target block sizes. More image parts coded at a higher coding level will result in a lower bit rate. Removal of affine-block-wise self-similarity is proposed which includes block-wise self-similarity as a special case. With the utilisation of affine-block-wise self-similarity, the library is substantially enriched which results in a higher probability of coding a target block at a higher coding level.

A very fast multi-level fractal block coding scheme exploiting affine-block-wise self-similarities is proposed. In the fast coding scheme, self-similarity in the very local area of the target block to be coded is exploited. By using affine-block-wise self-similarity, local correlations are exploited to a much further extent. The number of library blocks used for coding a target block is substantially reduced which results in very fast coding scheme. The proposed fast coding scheme outperforms previous implementations of the fractal block coding technique.

A hybrid fractal block coding and DCT scheme is proposed which codes a subsampled image using fractal block coding techniques. The fractal codes are used to decode by zooming to the original image size. The DCT technique is introduced to code the residue image. The proposed scheme is better than the pure fractal block coding scheme. The advanced fractal block coding schemes and the hybrid coder for still images are also applied to video compression which also give some promising simulation results.

INDEX TO FIGURES

Chapter 1.

Introduction

Image compression is a rapidly evolving field with growing applications in science and engineering. It is concerned with minimizing the number of bits required to represent an image. The need for image data compression arises from the amount of information contained in quality digital image data. Applications of data compression are primarily in transmission and storage of information. Image transmissions are in broadcast television, remote sensing via satellite, military communications via aircraft, radar and sonar, videotelephony, teleconferencing, computer communications facsimile transmission, and the like. Image storage is required for educational and business documents, medical images that arise in computer pictures, satellite images, weather maps, geological surveys, large image database and so on.

1.1 Image Raw Data Rates

Typical television images have a spatial resolution of approximately 512×512 pixels per frame. At 8 bits per pixel per colour channel and 30 frames per second, this translates into a rate of nearly 180×10^6 bits/s. Depending on the application, digital

image raw data rates can vary from 10^5 bits per frame to 10^8 bits per frame or higher. The large-channel capacity and memory requirement(See Table 1.1 & 1.2) for digital image transmission and storage make it desirable to consider data compression techniques.

Table 1.1 Data volumes of image sources(in Millions of Bytes)

Image Source	Data Volume (Megabytes)
National archives	12.5×10^9
1 hour of television	28×10^3
Encyclopedia Britannica	12.5×10^3
Book(200 pages of text)	1.3
one page viewed as an image	.13

Table 1.2 Storage capacities(in Millions of Bytes)

Media	Capacity (Megabytes)
Human brain	1,250,000,000
Magnetic cartridge	250,000
Optical disc memory	12,500
Magnetic disc	760
2400-ft magnetic tape	200
Floppy disc	1.25

1.2 Information Rates[89]

The raw image data rate does not however, necessarily represent its average information rate, which for a source with L possible independent symbols with probabilities p_i, $i = 0,1,2,...,L-1$, is given by entropy

$$H = -\sum_{i=0}^{L-1} p_i \log_2 p_i \quad \textit{bits per symbol}$$

According to Shannon's *noiseless coding theorem* [71], it is possible to code, without distortion, a source of entropy H bits per symbol using $H+\epsilon$ bits per symbol, where ϵ is an arbitrarily small positive quantity. Then the maximum achievable compression C, defined by

$$C = \frac{\textit{average bit rate of the original raw data}(B)}{\textit{average bit rate of the encoded data}\,(H+\epsilon)}$$

is $B/(H+\epsilon) \approx B/H$. Computation of such a compression ratio for images is impractical, if not impossible. For example an $N \times M$ digital image with B bits per pixel is one of $L = 2^{BNM}$ possible image patterns that could occur. Thus if p_i, the probability of the ith image pattern, were known, one could compute the entropy, that is, the information rate for B bits per pixel $N \times M$ images. Then one could store all the L possible image patterns and encode the image by its address - using a suitable encoding method, which would require approximately H bits per image or H/NM bits per pixel.

The entropy of an image can also be estimated from its conditional entropy. For a block of N $u_0, u_1,...,u_{N-1}$, with B bits per pixel and arranged in an arbitrary order, the Nth-order conditional entropy is defined as

$$H_N \triangleq -\sum_{u_0} \cdots \sum_{u_{N-1}} p(u_0,u_1,...,u_{N-1}) \log_2 p(u_0 | u_1,...,u_{N-1})$$

where each u_i, $i = 0,...,N-1$, takes 2^B values, and $p(\cdot,\cdot, ...)$ represent the relevant probabilities. For 8-bit monochrome television-quality images, the zero to second-order entropies (with nearest-neighbour ordering) generally lie in the range of 2 to 6 bits/pixel. Theoretically, for ergodic sequences,

as $N \rightarrow \infty$, H_N converges to H, the per-pixel entropy. Shannon's theory tells us that the bit rate of any exact coding method can never be below the entropy H.

1.3 Image Coding Environment

A typical environment for image coding is shown in Figure 1-1. The digital image is encoded by an image coder, also referred to as a *source coder*. The output of the image coder is a string of bits that represents the source image. The channel coder transforms the string of bits to a form suitable for transmission over a communication channel through some form of modulation. The modulated signal is then transmitted over a communication channel. The communication channel typically introduces some noise, and provision for error correction is usually made in the channel coder to

compensate for this channel noise. At the receiver, the received signal is demodulated and transformed back into a string of bits by a channel decoder. The image decoder reconstructs the image from the string of bits. For human viewing, the reconstructed image is typically displayed on a CRT monitor or printed out. In contrast to the communication environment in Figure 1-1, no communication channel is involved in applications of image coding for storage reduction. In storage reduction applications, the string of bits from the image coder is stored in proper format on a recording medium, ready for future retrieval. In this dissertation we are mainly concerned with the coder and decoder.

The image coder in Figure 1-1 consists of three elements, shown in Figure 1-2. The first and most important element is the transformation of the image to the most suitable domain for quantization and codeword assignment. In essence, this element determines what specifically is coded. Image coding algorithm classification is introduced in the following section. The second element in the image coder is quantization. To represent an image with a finite number of bits, image intensities, transform coefficients, or model parameters must be quantized. Quantization involves assignment of the quantization (reconstruction) levels and decision boundaries. The third element in the image coder is assignment of codewords, the strings of bits that represent the quantization levels.

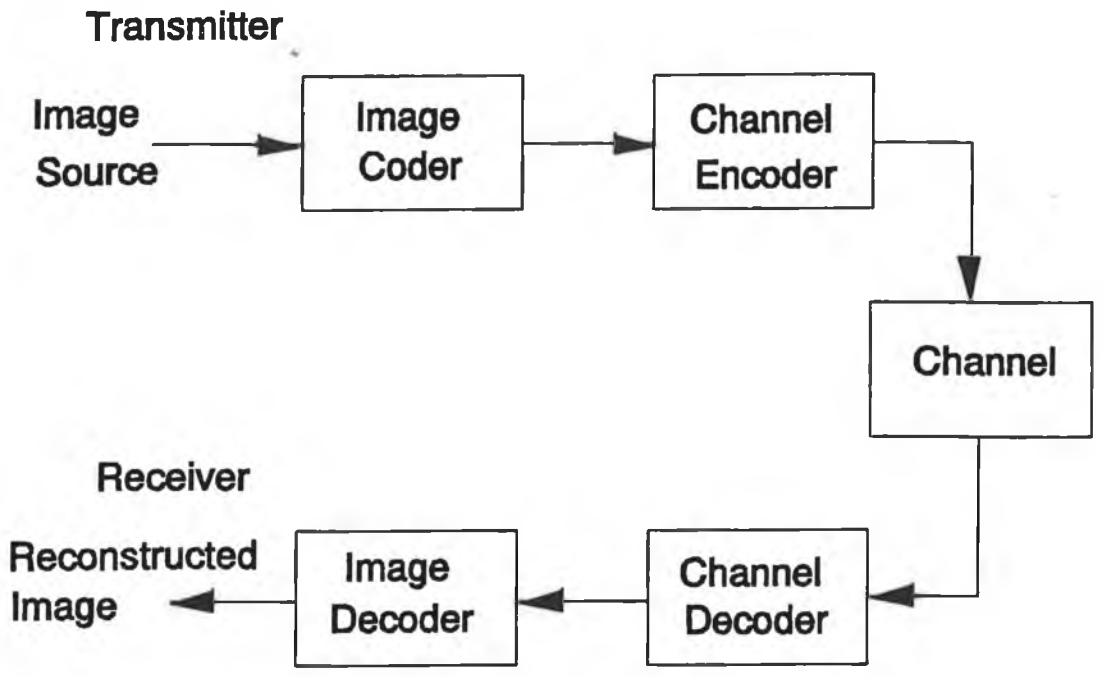


Fig. 1-1 Typical environment for image coding

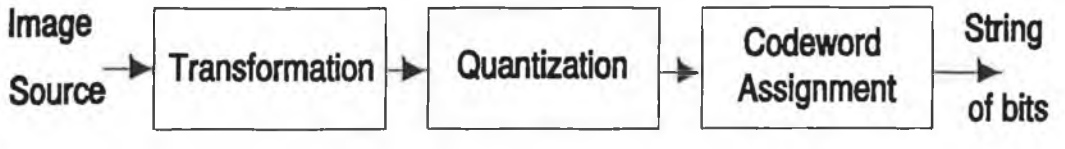


Fig. 1-2 Three major components in image coding

Each of these elements attempts to exploit the redundancy present in the image source and the limitations of the display device and the human visual system. As a result, the elements are closely interrelated.

1.4 Classification of Coding Algorithms

Image data compression methods can be broadly classified into four main categories which are *waveform coding, transform coding, image model coding, fractal image coding.* Other coding algorithms are mainly a hybrid based on these main categories. The performance and complexity of coding algorithm vary greatly, depending on what specifically is coded. We will briefly introduce each of the coding methods.

1.4.1 Waveform Coding

In waveform coding, we code the image intensity itself or some simple variation of image intensity such as the difference between two consecutive pixel intensities. One major advantage of waveform coding over transform coding and other coding methods is its simplicity. Since the waveform itself is coded, the coders are very simple both conceptually and computationally. Waveform coding techniques attempt to reconstruct

the waveform faithfully without detailed exploitation of information specific to a particular class of signals, and can therefore be used in coding many different classes of signals, such as images and speech. In addition, the bit rate reduction that can be achieved by waveform coders is comparable to that of transform coders in applications such as digital television, where high quality images are required. In applications such as video conferencing and remotely piloted vehicles, where bit rate reduction is important and some sacrifice in quality can be tolerated, waveform coders do not generally perform as well as transform coders.

A coding of this kind is *differential pulse code modulation*, or **DPCM** [31,32]. We assume that any adjacent pixel will tend to be near in brightness to its horizontal neighbour. As we travel across the line, the coded image file is loaded with the difference in brightness from one pixel to the next. Assuming that this change is never to be more than, say, eight grey levels, the coded change value need only be three bits long. Instead of storing the absolute brightness value of all pixels in an image, we simply store the change in brightness as we go from one pixel to the next.

An artifact of this coding technique shows up when the change in brightness from one pixel to the next is larger than eight grey levels. The **DPCM** method may require several pixel distances to settle on the correct pixel brightness value after encountering a sharp change. The effect of this phenomenon manifests itself as an apparent horizontal low pass filtering of the image. By increasing the number of bits used to

describe the pixel-to-pixel difference, this effect may be minimized. However, the improvement comes at the cost of data reduction efficiency.

1.4.2 Transform Coding

In *transform coding*, an image is transformed to a domain significantly different from the image intensity domain, and the transform coefficients are then coded. Transform coding techniques attempt to reduce the correlation that exists among image pixel intensities more fully than do waveform coding techniques. When the correlation is reduced, redundant information does not have to be coded repeatedly. Transform coding techniques also exploit the observation that for typical images a large amount of energy is concentrated in a small fraction of the transform coefficients. This is called the *energy compaction property*. Because of this property, it is possible to code only a fraction of the transform coefficients without seriously affecting the image. This allows us to code images at bit rates below 1 bit/pixel with a relatively small sacrifice in image quality and intelligibility.

Among the existing transform based methods, those based on the DCT [33,34,35] are the most widely used, both because of the decorrelative property of the DCT and the existence of fast algorithms for computing the DCT, implemented in VLSI. However, DCT coding methods suffer from the fact that the DCT is not local in the spatial

domain. To take advantage of the local spatial correlations in images, the DCT must then be calculated over a number of separate regions, which leads to blocking effects for very low bit rate transmission.

1.4.3 Image Model Coding

In *image model coding*, an image or some portion of an image is modelled and the model parameters are used for reconstructing the image. An image model coder can be viewed as an analysis/synthesis system. At the transmitter, the model parameters are estimated from analyzing the image, and at the receiver the image is synthesized from the estimated and quantized model parameters.

Image model coders [44-49] have the potential to synthesize intelligible images at a bit rate substantially lower than that necessary for waveform or transform coders. However, they are still at the research stage, and much needs to be done before their use becomes feasible in very low bit rate applications, such as video telephones. Developing a simple model capable of synthesizing intelligible images is not an easy problem. In addition, estimating model parameters and synthesizing an image from them are likely to be very expensive computationally.

1.4.4 Fractal Image Coding

Fractal image coding has been attracting a lot of interest in recent years for low-bit-rate image coding. The term 'fractal' was first coined by mathematician Benoit Mandelbrot in 1977. His later book [1] has captured the scientific imagination, giving an insight into many natural phenomena, from biology through physics to geology and much more. Scientists have found that typical fractal shapes exhibit self-similarity. Fractals are interesting because the images they generate are complex, can be magnified ad infinitum, and are often very eye-catching.

Fractals are discovered as the fixed points of certain set maps [2]. They are generated by the application of simple transformations on simple spaces. With Barnsley's IFS(iterated function system)[Chapter 2], we can generate fractal images with infinite details. The inverse problem is how to find IFS codes which can generate a desired image. Fractal image compression was first proposed by Barnsley and Sloan in [4]. According to Barnsley's collage theorem with iterated function system[Chapter 2], it should be possible to find IFS codes to encode any given black-and-white images.

In Barnsley's fractal image compression, a digitized image is first broken up into segments by image-processing techniques such as colour separation, edge detection, spectrum analysis, and texture-variation analysis. They then look up these segments

in a library of fractals. The library doesn't contain literal fractals; that would require astronomical amounts of storage. Instead, their library contains relatively compact sets of numbers which are IFS codes, that will reproduce the corresponding fractals. Furthermore, the library's cataloging system is such that images that look alike are close together: nearby codes correspond to nearby fractals. This makes it feasible to set up automated procedures for searching the library to find fractals that approximate a given target image. Once all the segments in the library are searched and IFS codes are found, the original digitized image could be thrown away. The codes are kept to retrieve the original image.

Jacquin [9-13] has proposed *fractal block* coding which relies on the assumption that image redundancy can be exploited through *block self-transformability*. Arbitrary digital greytone images are modelled as fixed points of contractive transformations on the image space. The transformation coefficients are used to iterate successively from any initial image to a fixed point which is close to the original image. Fractal block coding is typically asymmetric. Coding is much more computationally costly than decoding which results from the search effort while coding a target block. The computational complexity of fractal block coding scheme is a major impedient for real-time application. Beaumont [14-15] has investigated several implementation options to improve coding efficiency. Øien, et al [20] proposed an improved technique which finds the part of transformation coefficients by an optimisation technique to improve coding efficiency. Research in the fractal block coding technique is the main concern

of this dissertation.

Monro and Dudbridge [16] propose a fractal approximation of image blocks. This method uses system of affine transformations, which is a particular case of iterated function systems. A least-squares approximation to a grey-scale fragment is used, which is a particular solution to the inverse problem which seeks an IFS to describe a given fractal. A block is encoded without searching, by tiling it with reduced copies of itself using a least-squares criterion to derive an optimal mapping. Previously known matching techniques used in fractal block coding are subsets of this generalized method, which is called the **Bath Fractal Transform** (BFT). By introducing searching for the best image region for application of the BFT, a hybrid of known methods [17] is achieved. The efficiency of this method suggests that it is feasible for real time applications such as fractal video. The method readily extends to data of higher or lower dimensions, including time as in image sequences.

1.5 Dissertation Layout

In chapter 2, we introduce the mathematical foundation of fractal image coding. Transformations on metric spaces are introduced. Particular attention is given to contractive mapping on fractal spaces. It is explained what an iterated function system(IFS) is, and how it can define a deterministic fractal. We will give some

examples to show Barnsley's image coding idea with the help of the collage theorem with the IFS. The optimisation problem of finding the minimum number of transformations which gives reasonably good cover is presented. Finally we apply the collage theorem with the IFS to image space which is the foundation theorem for the fractal block coding technique.

In chapter 3, we will study the fractal block coding scheme proposed by Jacquin [9-13]. Simulations based on Jacquin's fractal block coding scheme show that block-wise self-similarity in a natural image is not fully exploited while keeping the size of the target block size fixed. A multi-level fractal block coding scheme is proposed in this dissertation. The proposed scheme combines different target block size levels to further exploit self-similarity within an image. This enables the coder to code an image with different target blocks adaptively according to the level of complexity of the image.

To achieve a low bit rate, we want more blocks to be coded at a higher coding level. We propose a new library strategy removing affine-block-wise self-similarity which includes block-wise self-similarity as a special case. With the new library strategy, a target block has a better chance to be coded at a higher target block size level. Hence, a low bit rate can be achieved with good subjective image quality.

Finally we propose a very fast coding scheme which substantially increases coding

efficiency. We abandon the library block search effort adopted in previous implementations of fractal block coding[9-13,20]. Instead of searching through the image library blocks, we just use the blocks that covers the target block as our library block. The library block covering the target block frequently fails to provide a good match which results in searching for a library block somewhere further from the target block in the original image. However with affine-block-wise self-similarity, we can make full use of local library block information to give us a good match. Since there is no need to search for a library block in other parts of the image, bits previously used for the address of the library block are saved. Therefore, this new coding scheme not only drastically speeds up coding, but also brings down the bit rate for each target block. Simulations have demonstrated that the new coding scheme is a very promising scheme which brings the fractal block coding technique closer to practical realtime application.

In chapter 4, the relation between image resolution and target block size is first exploited. Coding a higher resolution image with a big target block size is equivalent to coding a smaller resolution with a smaller target block size. This suggests a more efficient way of coding higher resolution images. By combining fractal zooming with the DCT, we propose an efficient codec for still image coding which can have a direct application in the contemporary idea of layered coding to suit different bandwidth requirements.

Chapter 5 introduces the application of the still image codec presented in chapter 3 and 4 to video compression application. Good simulation results are achieved which suggests fractal block coding technique may be useful in the area of video compression. The motion prediction technique is found to be not practical for fractal block coding.

Chapter 6 is the conclusion and discussion. The research achievement in fractal block coding is summarised. Several schemes are proposed to improve the coding efficiency further. We also discuss the possible direct combination with the object-oriented analysis-synthesis codec [34,35] which is a very promising video codec for very low bit rates.

Chapter 2

Mathematical Background

2.1 Introduction

In fractal image coding, arbitrary greytone images are modelled as fixed points of contractive transformations on the image space. The transformation coefficients are used to iterate successively from any initial image to a fixed point which is an approximation to the original image. The number of bits used for the transformations is much lower than that of the original image. The mathematical foundation of this coding technique is rooted in iterated transformation theory. Thus it is essential for us to have some basic knowledge of iterated transformation theory and some crucial mathematical theorems which theoretically guarantee that the idea of image coding by contractive transformation is achievable and practical. We will introduce the metric space where fractals live. We will then lead the readers to the world of deterministic fractals. Since fractal block coding is an extension of IFS(iterated function system) image encoding, iterated function system image encoding is introduced in depth. Finally we will introduce the collage theorem on image spaces which is the most important foundation theorem for solving the inverse problem of image coding of natural scenes. Most of the important definitions and basic theorems are presented

while only proofs for the most important theorems are provided. The following sections are based on Hutchinson[3] and Barnsley[2].

2.2 Metric Spaces

2.2.1 Space and Metric Spaces

In fractal geometry we are dealing with the structures of subsets of different very simple "geometrical" spaces. Such a space is denoted by $\mathcal{P}$. It is the space on which we think of drawing our fractals; it is the space where fractals live. "Fractal" for us at the moment is simply a subset of a space. Whereas the space is simple, the fractal subsets may be geometrically complicated.

Definition 2.1

A space $\mathcal{P}$ is a set. The *points* of the space are the elements of the set.

Example 1: $\mathcal{P} = \mathbf{R}^2$. the Euclidean plane, the coordinate plane of calculus. Any pair of real numbers $p_1, p_2 \in \mathbf{R}$ determine a single point in $\mathbf{R}^2$.

Example 2: $\mathcal{P}$=C[0,1], the set of continuous functions which take the real closed interval [0,1] into $\mathbf{R}$.

This definition does not imply the spatial relation of two points in the sense of closeness. We have to quantify the notion of *closeness* and *difference* by introducing a measure so as to distinguish between different points.

Definition 2.2

Let $\mathcal{P}$ be a space. The function $d: \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}^+$ is a *metric* if it satisfies the following conditions:

i) $\forall$ a,b $\in \mathcal{P}$, d(a,b) = 0 if and only if a = b.

ii) $\forall$ a,b $\in \mathcal{P}$, d(a,b) = d(b,a) (symmetry).

iii) $\forall$ a,b,c $\in \mathcal{P}$, d(a,c) $\leq$ d(a,b) + d(b,c) (triangular inequality).

Definition 2.3

A space $\mathcal{P}$ with a metric d is called metric space which is denoted by ($\mathcal{P}$,d).

Here is an example of metric space.

$\mathcal{P} = \mathbb{R}^2$

$$d_1 = \sqrt{(x_1 - y_1)^2 - (x_2 - y_2)^2}$$

$$d_2 = \|x_1 - y_1\| + \|x_2 - y_2\|$$

Both $(\mathcal{P},d_1)$ $(\mathcal{P},d_2)$ are metric space.

2.2.2 Sequences, Convergence, Completeness

A finite sequence $(x_0,x_1,x_2,...,x_{N-1})$ of space $\mathcal{P}$ can be expressed as $\{x_n;n=0,1,2,...,N\text{-}1\}$. The index n is sometimes called the rank of x_n. An infinite sequence of points of $\mathcal{P}$ can be expressed as $\{x_n;n=0,1,2,3,...,\infty\}$. We use "sequence" to mean either a finite or an infinite sequence. We now introduce some notions which relate to the possible *clustering* of objects in an infinite sequence expressed as $\{x_n\}$ for short.

Definition 2.4

A sequence $\{x_n\}$ in a metric space $(\mathcal{P},d)$ is called a *Cauchy sequence* if, for any given number $\epsilon > 0$, there is an integer $N > 0$ so that

$$d(x_n,x_m) < \epsilon \quad \text{for all } n,m > N.$$

In other words, the further along the sequence one goes, the closer together become the points in the sequence. Although points are getting closer, it does not mean points are approaching a point.

Example 2: (C[0,1],D) is a complete metric space, where D is defined by

$$D(f,s) = \text{Max}(|f(s) - g(s)| : s \in [0,1]).$$

The sequence we are interested in this thesis has the form of $\{\tau^n(x)\}$ where x is some initial point of $\mathcal{P}$ and τ is a mapping from $\mathcal{P}$ to $\mathcal{P}$. We call this kind of sequence an iterated sequence and call τ the iterated transformation.

Definition 2.7

Let $S \subset \mathcal{P}$ be a subset of metric space $(\mathcal{P},d)$. S is *compact* if every infinite sequence $\{x_n\}$ in S contains a subsequence having a limit in S.

2.2.3 Space of Fractals

We come to the ideal space in which to study fractal geometry. To start with, and always at the deepest level, we work in some complete metric space, such as ($\mathbb{R}^2$, Euclidean), which we denote by $(\mathcal{P},d)$. But then, when we wish to discuss pictures, drawings, "black-on-white" subsets of the space, it becomes natural to introduce the space $\mathcal{H}$.

Definition 2.8

Let $(\mathcal{P},d)$ be a complete metric space. The $\mathcal{H}(\mathcal{P})$ denotes the space whose points are the compact subsets of $\mathcal{P}$, other than the empty set.

Definition 2.9

Let $(\mathcal{P},d)$ be a complete metric space, $x \in \mathcal{P}$, and $B \in \mathcal{H}(\mathcal{P})$.

Define

$$d(x,B) = \text{Min}\{d(x,y): y \in B\}.$$

Then $d(x,B)$ is the distance from the point x to the set B.

Definition 2.10

Let $(\mathcal{P},d)$ be a complete metric space. Let $A,B \in \mathcal{H}(\mathcal{P})$.

Define

$$d(A,B)=\text{Max}\{d(x,B): x \in A\}.$$

$d(A,B)$ is the *distance* from the set A and $B \in \mathcal{H}(\mathcal{P})$.

Definition 2.11

Let $(\mathcal{P},d)$ be a complete metric space, then the *Hausdorff distance* between points A

and B in $\mathcal{H}(\mathcal{P})$ is defined by

$$h(A,B) = \text{Max}\ \{\ d(A,B),\ d(B,A)$$

We refer to $(\mathcal{H}(\mathcal{P}),h)$ as "the space of fractal". It is too soon to be formal about the exact meaning of "a fractal". At the present stage of development of science and mathematics, the idea of a fractal is most useful as a broad concept. Fractals are not defined by a short legalistic statement, but by the many pictures and contexts which refer to them. For us any subset of $(\mathcal{H}(\mathcal{P}),h)$ is a fractal. However, as with the concept of "a space", more meaning is suggested than is formalized.

Theorem 2.2(The Completeness of the Space of Fractals)

Let $(\mathcal{P},d)$ be a complete metric space. Then $(\mathcal{H}(\mathcal{P}),h)$ is a complete metric space. Moreover, if $\{A_n \in \mathcal{H}(\mathcal{P})\}$ is a Cauchy sequence then

$$A = \lim_{n\to\infty} A_n \quad \in \mathcal{H}(\mathcal{P})$$

The above theorem is a very important theorem. Space of fractals $(\mathcal{H}(\mathcal{P}),h)$ is complete metric space as long as $(\mathcal{P},d)$ is a complete space. Therefore we have the space of fractals we are interested in $(\mathcal{H}(\mathbb{R}^2),h)$ is a complete space.

2.3 Transformations on Metric Spaces

Fractal geometry studies "complicated" subsets of geometrically "simple" spaces such

as $\mathbb{R}^2$,$\mathbb{R}$,$\mathbb{C}$. In deterministic fractal geometry the focus is on those subsets of a space which are generated by , or possess invariance properties under, simple geometrical transformations of the space into itself. A simple geometrical transformation is one which is easily conveyed or explained to someone else. Usually they can be completely specified by a small set of parameters. Examples include affine transformations in $\mathbb{R}^2$, which are expressed using 2×2 matrices and 2 vectors.

2.3.1 Contractive Transformations

Definition 2.11

Let $(\mathcal{P},d)$ be a metric space. A *transformation* on $\mathcal{P}$ is a function

$$\tau: \mathcal{P} \to \mathcal{P},$$

which assigns exactly one point $\tau(x) \in \mathcal{P}$ to each point $x \in \mathcal{P}$.

If $S \subset X$ then $\tau(S) = \{\tau(x): x \in S\}$.

τ is *one-to-one* if $x,y \in \mathcal{P}$ with $\tau(x) = \tau(y)$ implies $x = y$.

Function τ is *onto* if $\tau(\mathcal{P}) = \mathcal{P}$.

τ is called *invertible* if it is one-to-one and onto: in this case it is possible to define a transformation $\tau: \mathcal{P} \to \mathcal{P}$ called the inverse of τ, by $\tau^{-1}(y) = x$ where $x \in \mathcal{P}$ is the unique point such that $y = \tau(x)$.

Definition 2.12

Let $\tau: \mathcal{P} \to \mathcal{P}$ be a transformation on a metric space. The *forward iterates* of τ are transformations

$$\tau^{o0}(x) = x\ ,\ \tau^{o1} = \tau(x)\ ,\ \ldots,\ \tau^{o(n+1)} = \tau\circ\tau^{on} = \tau(\tau^{on}) \quad \textit{for } n=0,1,2,\ldots$$
$$x_0 = x\ ,\ x_1 = \tau(x_0) = \ldots,\ x_{n+1} = \tau(x_n)$$

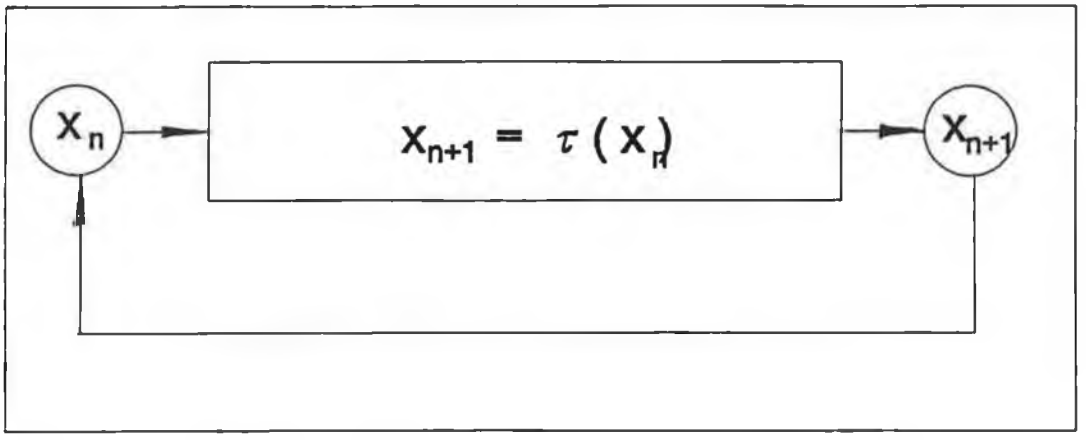


Fig. 2-1 One step iteration

In order to work in fractal geometry one needs to be familiar with the basic families of transformations in $\mathbb{R}$, $\mathbb{R}^2$, and $\mathbb{C}$. One needs to know well the relationship between "formulas" for transformations and the geometric changes, stretchings, foldings, and skewings of the underlying fabric, the metric space upon which they act. It is more

important to understand what the transformations do to sets than how they act on individual points. So, for example, it is more useful to know how an affine transformation in $\mathbb{R}^2$ acts on a straight line, a circle, or a triangle, than to know where it takes the origin.

When the transformation τ on an iterated sequence contracts distances, much can be said about the behaviour of the sequence. We need the following definitions.

Definition 2.13

A transformation $\tau: \mathcal{P} \rightarrow \mathcal{P}$ on a metric space $(\mathcal{P},d)$ is called *contractive or a contraction mapping* if there is a constant $0 \leq s < 1$ such that

$$d(\tau(x),\tau d(y)) \leq s \cdot d(x,y) \quad \forall\, x,y \in \mathcal{P}$$

Any such number s is called a *contractive factor* for τ.

Definition 2.14

Let τ be a transformation in $\mathcal{P}$. A *fixed point* of τ is a point x' in $\mathcal{P}$ that is invariant under application of τ: i.e such

$$\tau(x') = x'.$$

Theorem 2.3 [The Contraction Mapping Theorem]

Let $\tau: \mathcal{P} \to \mathcal{P}$ be a contraction mapping on a complete metric space $(\mathcal{P},d)$, then τ possesses exactly one fixed point $x_f \in \mathcal{P}$ and moreover for any point $x \in \mathcal{P}$, the sequence $\{\tau^{on}(x): n = 0,1,2,...\}$ converges to x_f. That is

$$\lim_{n\to\infty} \tau^{on}(x) = x_f \quad \forall\, x \in \mathcal{P}$$

Proof. Let $x \in \mathcal{P}$. Let $0 \le s < 1$ be a contractivity factor for τ. Then

$$d(\tau^{on}(x),\tau^{om}(x)) \le s^{\min(m,n)} d(x,\tau^{o|m-n|}(x))$$
$$\textit{for all } m,n = 0,1,2,...,$$

we have

$$\begin{aligned} d(x,\tau^{ok}(x)) &\le d(x,\tau(x)) + d(\tau(x),\tau^{o2}(x)) + ... + d(\tau^{k-1}(x),\tau^{ok}(x)) \\ &\le (1+s+s^2+...+s^{k-1}) d(x,\tau(x)) \\ &\le (1-s)^{-1} d(x,\tau(x)) \end{aligned}$$

So substituting into the previous equation we now obtain

$$d(\tau^{om}(x),\tau^{on}(x)) \leq \frac{s^{\min(m,n)}}{1-s} d(x,\tau(x))$$

from which it immediately follows that $\{f^{on}(x)\}$ is a Cauchy sequence.

Since X is complete, this Cauchy sequence possesses a limit $x_f \in X$, and we have

$$\lim_{n\to\infty} \tau^{on}(x) = x_f .$$

Now we shall show that x_f is a fixed point of τ.

Since τ is contractive it is continuous and hence

$$\tau(x_f) = \tau(\lim_{n\to\infty} \tau^{on}(x)) = \lim_{n\to\infty} \tau^{o(n+1)}(x) = x_f .$$

Finally, can there be more than one fixed point? Suppose there are. Let x_f and y_f be two fixed points of τ. Then $x_f = \tau(x_f)$, $y_f = \tau(x_f)$, and

$$d(x_f,y_f) = d(\tau(x_f),\tau(y_f)) \leq s\cdot d(x_f,y_f)$$

whence $(1-s)d(x_f,y_f) \leq 0$, which implies $d(x_f,y_f) = 0$ and hence $x_f = y_f$.

This completes the proof.

2.3.2 Contraction Mappings on the Space of Fractals

Definition 2.15

A (hyperbolic) *iterated function system* consists of a complete metric space $(\mathcal{P},d)$ together with a finite set of contraction mappings $w_n: \mathcal{P} \to \mathcal{P}$, with respective contractivity factors s_n, for n= 1, 2, 3,...,N. The abbreviation "IFS" is used for "iterated function system." The notation for the IFS just announced is $\{\mathcal{P}: w_n, n=1,2,...,N\}$ and its contractivity factor is

$$s = \text{Max}\{s_n: n=1,2,...,N\}.$$

Theorem 2.4

Let $\{\mathcal{P}: w_n, n=1,2,...,N\}$ be a *hyperbolic iterated function system with contractivity factor s*. The transformation $W: \mathcal{H}(\mathcal{P}) \to \mathcal{H}(\mathcal{P})$ defined by

$$W(B) = \bigcup_{n=1}^{N} w_n(B)$$

for all $B \in \mathcal{H}(\mathcal{P})$, is a contraction mapping on the complete metric space $(\mathcal{H}(\mathcal{P}),h(d))$

with contractivity factor s. That is

$$h(W(B),W(C)) \leq s \cdot h(B,C)$$

for all B,C $\in \mathcal{H}(\mathbb{P})$. Its unique fixed point, A $\in \mathcal{H}(\mathbb{P})$, obeys

$$A = W(A) = \bigcup_{n=1}^{N} w_n(A)$$

and is given by $A = \lim_{n\to\infty} W^{\circ n}(B)$ for any B $\in \mathcal{H}(\mathbb{P})$.

Definition 2.16

The fixed point $A \in \mathcal{H}(\mathbb{P})$ described in the theorem is called the *attractor* of the IFS.

2.3.3 Affine Transformations in Euclidean Plane

Definition 2.17

A transformation w: $\mathbb{R}^2 \to \mathbb{R}^2$ of the form

$$w(x1,x2) = \{ ax_1 + bx_2 + e,\ cx_1 + dx_2 + f \}$$

where a,b,c,d,e and f are real constants, is called a(two-dimensional) *affine* transformation.

We can also express the transformation in the following matrix form which will give us better understanding of its function in $\mathbb{R}^2$.

$$w(x1,x2) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = Ax + t.$$

$$where\ A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad t = \begin{pmatrix} e \\ f \end{pmatrix}$$

Here A is a two-dimensional 2×2 real matrix and t is the column vector which we do not distinguish from the coordinate pair $(e,f) \in \mathbb{R}^2$. Such transformations have important geometrical and algebraic properties.

The matrix A can always be written in the form

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} l\cos(\alpha) & -r\sin(\beta) \\ l\sin(\alpha) & r\cos(\beta) \end{pmatrix}$$

where l is the scaling factor on x, r is the scaling factor on y, α is the angle of rotation on x, and β is the angle of rotation on y. e is the translation on x, and f is the

translation on y.

Such a representation is always possible by setting:

$$l = \sqrt{a^2 + c^2} \qquad \alpha = \arctan\left(\frac{c}{a}\right)$$

$$r = \sqrt{b^2 + d^2} \qquad \beta = \arctan\left(-\frac{b}{d}\right)$$

i.e (l,α) are the polar coordinates of the point (a,c) and $(r,(\beta+\pi/2))$ are the polar coordinates of the point (b,d). The linear transformation

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

in $\mathbb{R}^2$ maps any parallelogram with a vertex at the origin to another parallelogram with a vertex at the origin.

The general affine transformation $w(x) = Ax+t$ in $\mathbb{R}^2$ consists of a linear transformation, A, which deforms space relative to the origin, as described above, followed by a *translation* or *shift* specified by the vector t where e is the translation on x, and f is the translation on y.

Definition 2.18

A transformation w: $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ is called a *similitude* if it is an affine transformation having one of the special forms

$$w\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{pmatrix}r\cos(\alpha) & -r\sin(\alpha)\\ r\sin(\alpha) & r\cos(\alpha)\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix} + \begin{pmatrix}e\\f\end{pmatrix}$$

$$w\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{pmatrix}r\cos(\alpha) & r\sin(\alpha)\\ r\sin(\alpha) & -r\cos(\alpha)\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix} + \begin{pmatrix}e\\f\end{pmatrix}$$

for some translation $(e,f) \in \mathbb{R}^2$, some real number $r \neq 0$, and some angle α, $0 \leq \alpha < 2\pi$. α is called the rotation angle while r is called the *scale factor* or *scaling*.

The linear transformation

$$R_\alpha\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{pmatrix}\cos(\alpha) & -\sin(\alpha)\\ \sin(\alpha) & \cos(\alpha)\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix}$$

is a *rotation*.

The linear transformation

$$R\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix}$$

is a *reflection*.

Definition 2.19

The affine transformation is specified by six real numbers. Given an affine transformation, one can always find a nonegative number s so that

$$\|d(x),d(y)\| \leq s \cdot \|x-y\| \quad \text{for } \forall x,y \in \mathbb{R}^2$$

We call the smallest number s so that this is true the *Lipschitz constant* for w.

Here

$$\|x\| = \sqrt{x_1^2 + x_2^2}$$

Such an affine transformation is called contractive if $s < 1$, and it is called a symmetry if

$$\|w(x)-w(y)\| = \|x - y\| \quad \text{for } \forall x,y \in \mathbb{R}^2$$

It is expansive if its *Lipschitz constant* is greater than one.

2.4 Deterministic Fractals Generated by IFS

So far we have discussed about all the basic knowledge about metric space and about fractal spaces. However not a single fractal is presented nor is a definition of fractals given.

According to mathematicians, the definition of a 'fractal' should be regarded in the same way as the biologist regards the definition of 'life'. There is no hard and fast definition, but just a list of properties characteristic of a living thing, independently of the environment. Most living things have most of the characteristics on the list, thought there are living objects that are exceptions to each of them. In the same way, it seems best to regard a fractal as a set that has certain properties, rather than to look for a precise definition which will almost certainly exclude some interesting cases. From the mathematician's point of view, this approach is no bad thing. It is difficult to avoid developing properties of dimension other than in a way that applies to 'fractal' and 'non-fractal' sets alike. For 'non-fractals', however, such properties are of little interest - they are generally almost obvious and could be obtained more easily by other methods.

After enjoying the beauty of deterministic fractals, you might watch at your surrounding nature with a totally different perspective and come to realise that fractal geometry is really the geometry of nature.

We restrict our attention to hyperbolic IFS of the form $\{\mathbb{R}^2; w_n; n=1,2,3,...,N\}$, where each w_n is an affine transformation

$$w_i(x) = w_i\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{pmatrix}a_i & b_i\\c_i & d_i\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix} + \begin{pmatrix}e_i\\f_i\end{pmatrix} = A_i x + t_i .$$

which takes $\mathbf{R}^2$ into $\mathbf{R}^2$, together with a set of probabilities

$$\{p_n;\ n=1,2,...N\}$$

where each $p_i > 0$ and

$$p_1+p_2+p_3+...+p_N = 1.$$

Definition 2.20

Let s_n denote the Lipschitz constant for w_n for each n=1,2,...,N. then we say the IFS code obeys the *average contractivity condition* if

$$s_1^{p_1} \cdot s_2^{p_2} \cdot s_3^{p_3} \cdots s_N^{p_N} < 1$$

Definition 2.21

An IFS code is a hyperbolic IFS

$$\{\mathbf{R}^2;\ w_n, p_n; n=1,2,...,N\}$$

such that the average contractivity condition is obeyed.

Since each w_i is decided by a group coefficients, the IFS code can be efficiently expressed in the following table form:

Table 2.1 IFS codes table format:

w	a	b	c	d	e	f	p
1	a_1	b_1	c_1	d_1	e_1	f_1	p_1
2	a_2	b_2	c_2	d_2	e_2	f_2	p_2
...	...	...	...	...			
N	a_N	b_N	c_N	d_N	e_N	f_N	p_N

There are two algorithms for the generation of IFS attractors. One is a random iteration algorithm and the other is a deterministic algorithm.

1. Random iteration algorithm

Let $\{\mathcal{P}; w_1,w_2,w_3,...,w_N\}$ be a hyperbolic IFS, where probability $p_i > 0$ has been assigned to w_i for i=1,2,...N, where

$$p_1+p_2+p_3+...+p_N = 1.$$

Choose $x_0 \in \mathcal{P}$ and then choose recursively, independently,

$$x_n \in \{ w_1(x_{n-1}),w_2(x_{n-1}),w_3(x_{n-1}),...,w_N(x_{n-1}) \}$$

$$for\ n=1,2,3,...,$$

where the probability of the event $x_n = w_i(x_n\text{-}1)$ is p_i. Thus construct a sequence $\{x_n;\ n=0,1,2,3,...\}$ contained in $\mathcal{P}$.

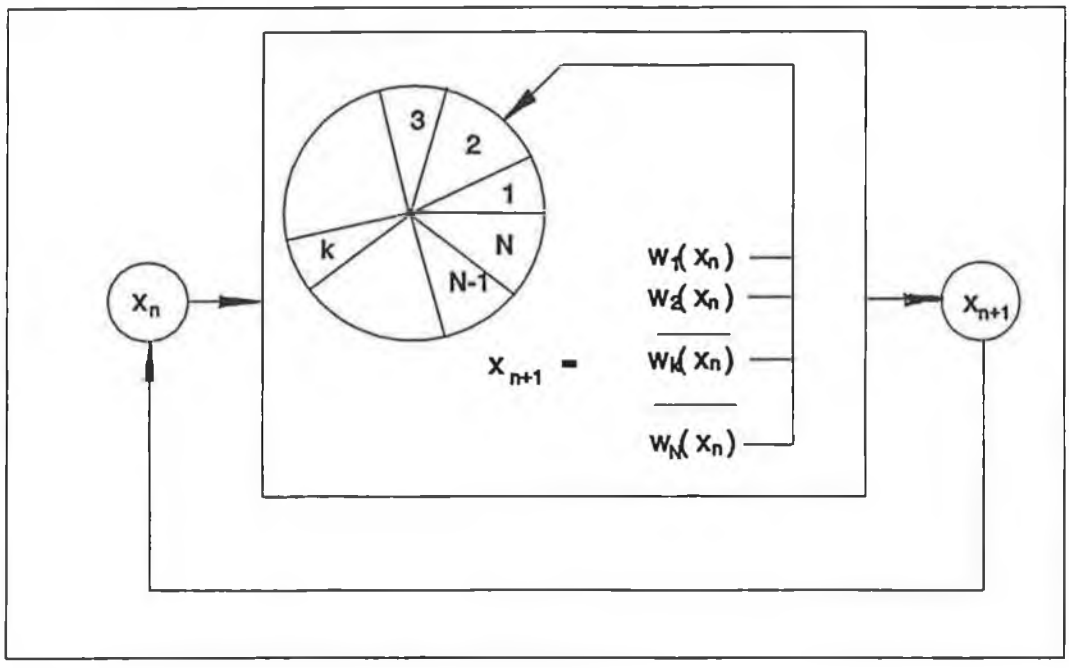


Fig.2-2 Feedback machine of fortune wheel

Different transformations are applied at different frequencies. If each affine transformation is applied at exactly equal probability, we will have the deterministic algorithm. This algorithm is graphically explained by Fig. 2-2.

2. Deterministic algorithm

Let $\{\mathbb{P}; w_1,w_2,w_3,...,w_N\}$ be a hyperbolic IFS. Choose a compact set $A_0 \in \mathbb{R}^2$.

Then compute successively $A^n = W_{on}(A)$ according to

$$A_{n+1} = \bigcup_{i=1}^{N} w_i(A_n) \quad for\ n=1,2,...$$

Thus construct a sequence $\{A_n\}$ which converges to the attractor of the IFS in the Hausdorff metric.

When the probabilities are distributed properly, the chaos algorithm has much faster convergence speed than deterministic algorithm. We will give some demonstrations of the deterministic algorithm. All fractals generated in this section are deterministic fractals.

The first deterministic fractal we are going to introduce here is the famous Sierpinski triangle. By applying the same IFS codes(Table 2.2) on two different initial images we get an exactly same attractor. The attractor is called the *Sierpinski triangle* which was named after the famous Polish mathematician Waclaw Sierpinski(1882-1969). We should notice that we can find copies of the whole Sierpinski triangle near every point of it. The triangle has intricate infinite detail. It is composed from small but exact copies of itself. So the Sierpinski triangle is called strictly self-similar. The attractor

is invariant under the IFS codes which generate it.

Figure 2-3 is an approximation of the Sierpinski triangle because of the limitation of our graphical resolution. A real Sierpinski should have infinite details. If you pick up any small triangle and zoom it up, it still has infinite details. There are infinitely many points and the organization is so complicated that there is no way we can describe this set by specifying each point. Instead it can better be expressed in terms of relations between pieces. Actually this Sierpinski triangle is generated by three affine transformations listed in Table 2-2. The iteration towards infinity shown in Figure 2-4 and Figure 2-5 gives us a Sierpinski triangle with infinite details.

Table 2.2 IFS code for Sierpinski triangle:

w	a	b	c	d	e	f	p
1	0.5	0	0	0.5	0	0	.333
2	0.5	0	0	0.5	.5	0	.333
3	0.5	0	0	0.5	.5	.5	.333

Table 2.3 IFS code for a Barnsley fern:

w	a	b	c	d	e	f
1	.849	.037	-.037	.489	.075	.183
2	.197	-.226	.226	.197	.400	.049
3	-.150	.283	.260	.237	.575	-.084
4	0.5	.000	.000	.160	.500	.000

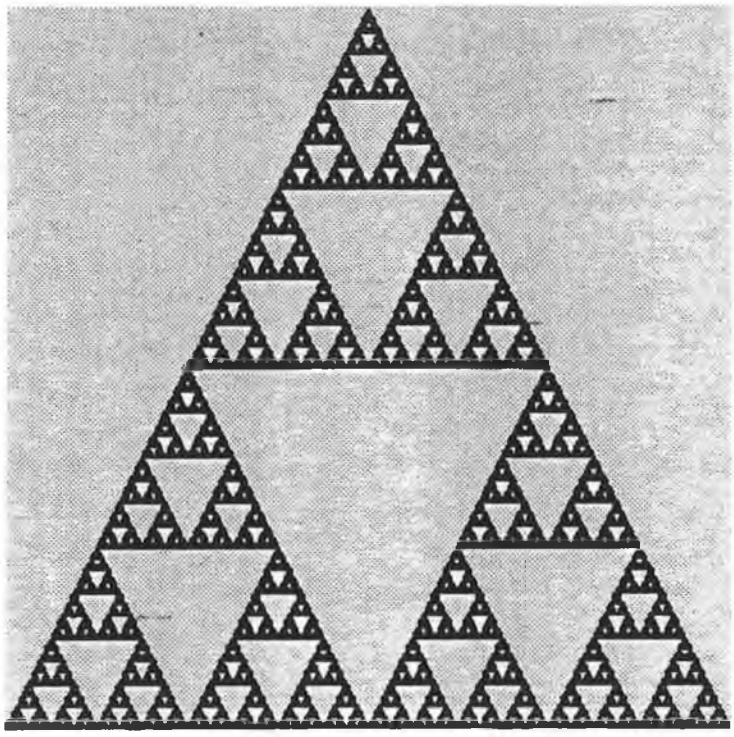


Fig. 2-3 Picture of the Sierpinski triangle

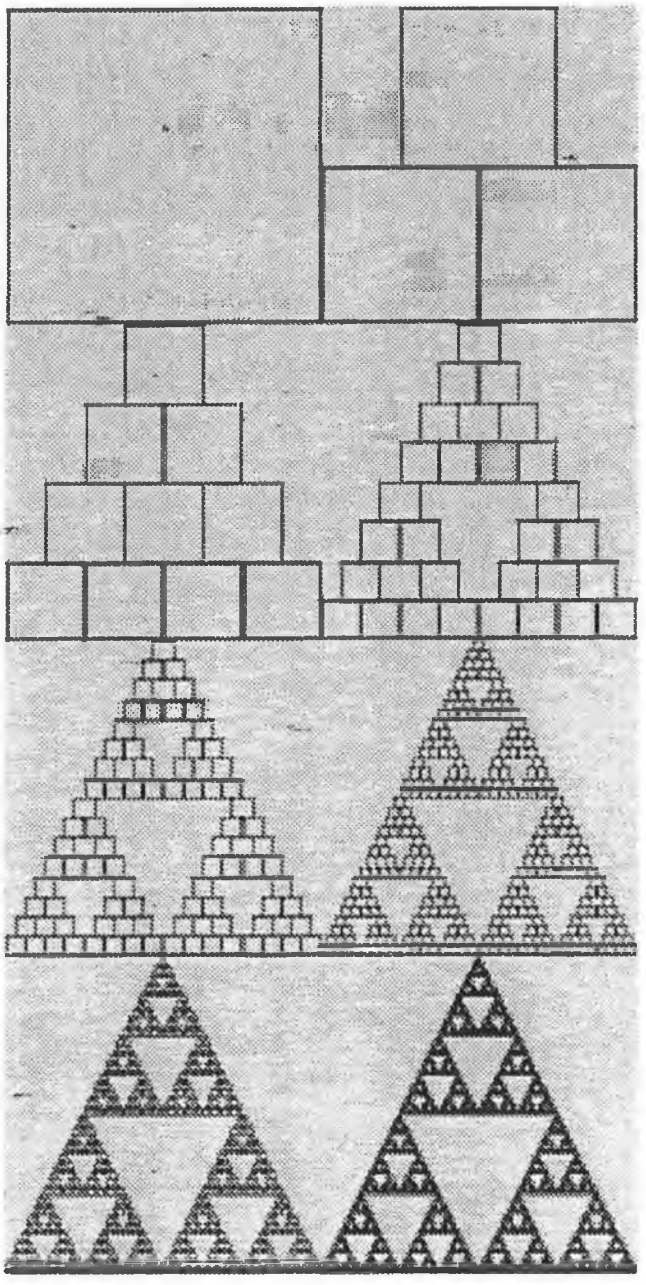


Fig. 2-4 Generation of Sierpinski triangle with square as starting point

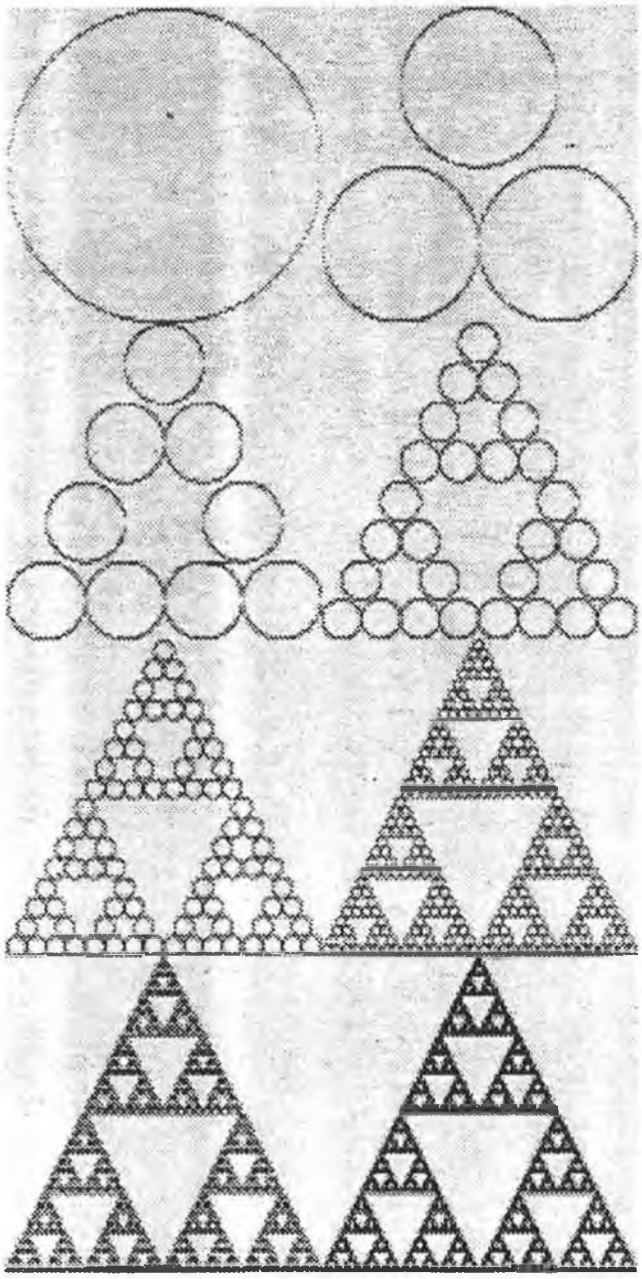


Fig. 2-5 Generation of the Sierpinski triangle with circle as starting point

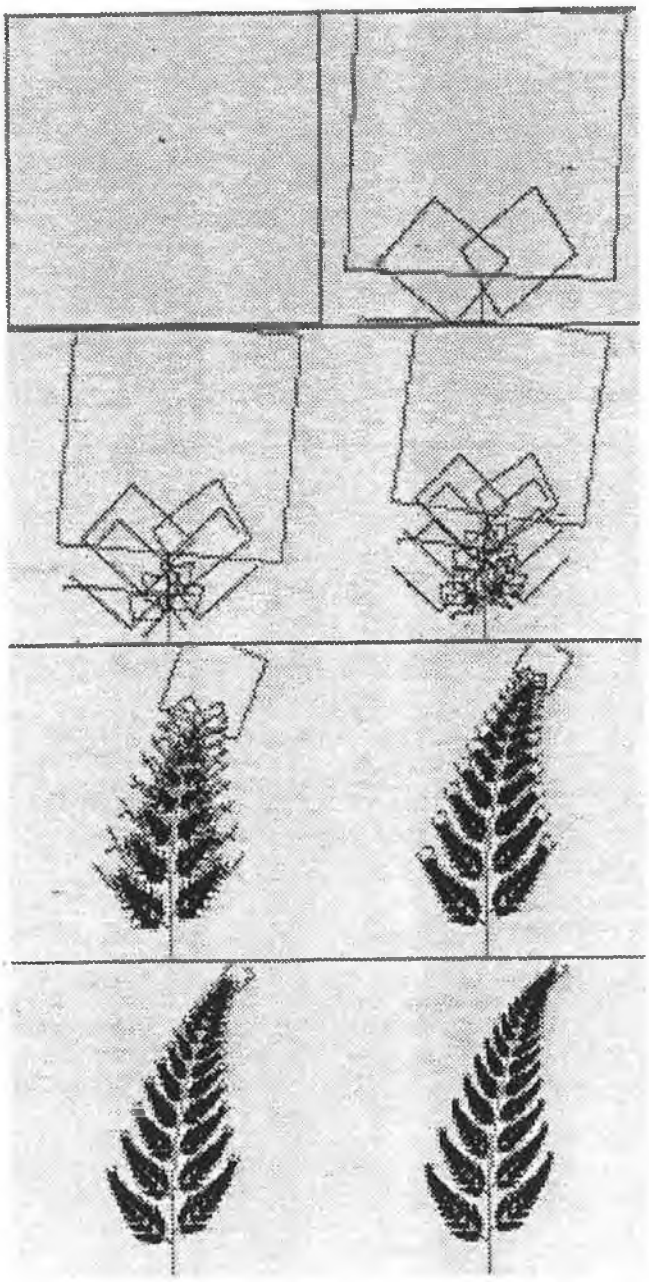


Fig. 2-6 Generation of a fern

Another deterministic fractal is the Barnsley fern which is shown in Figure 2-6 with IFS codes in Table 2.3.

In case of the fern, a smaller fern is an exact copy of the whole fern, however, that is not the case for the stem. A stem is not similar to the whole fern but we can interpret it as an affine copy which is compressed to a line. The attractor is not strictly self-similar but self-affine.

These demonstrations of deterministic fractals help us to understand the contraction mapping theorem more intuitively. For a hyperbolic IFS, the attractor is unique starting from any compact subset.

According to the deterministic fractals we have seen, we can summarize some properties of deterministic fractals as follows:

1.Has infinite intricate details at infinite magnifications.

2.Has self-similarity which means a fractal looks very much the same however much you magnify it.

3.A small group of fractal codes can generate geometrically complicated fractals. Conversely a geometrically complicated fractal image can be

compressed to a small amount of codes.

2.5 Solution of an Inverse Problem for Fractals

We have already noticed the amazing fact that a handful of IFS codes can generate geometrically complicated images with infinite details. In term of image data compression, these fractal IFS codes have substantially low information rate compared to the information they render. It is natural to ask whether it is possible to generate any desired image by IFS codes. If the IFS codes have a lower information rate, then we can reach our aim of image compression by keeping the codes instead of the original image. This inverse problem is partly solved by the important Collage theorem which we will introduce in depth in this section. The collage theorem provides us with general foundation of how to create the IFS for black-and-white natural images.

2.5.1 Collage Theorem

Theorem 2.5 (Collage theorem)

Let $(\mathcal{P},d)$ be a complete metric space. Let $\tau: \mathcal{P} \to \mathcal{P}$ be contraction mapping with contractivity factor $0 \leq s < 1$, and let the fixed point of τ be $x_f \in \mathcal{P}$.

Then

$$d(x,x_f) \leq \frac{d(x,\tau(x))}{1-s} \quad \text{for } \forall\, x \in \mathcal{P}$$

Proof:

The distance function $d(a,b)$ for fixed $a \in X$ is continuous in $b \in X$.

Hence

$$\begin{aligned} d(x,x_f) &= d(x,\lim_{n\to\infty}\tau^{on}(x)) \\ &= \lim_{n\to\infty} d(x,\tau^{on}(x)) \\ &= \lim_{n\to\infty}\sum_{m=1}^{n} d(\tau^{o(m-1)},\tau^{om}) \\ &\leq \lim_{n\to\infty} d(x,\tau(x))(1 + s + s^2 + \ldots + s^{n-1}) \\ &\leq \frac{d(x,\tau(x))}{1-s} \end{aligned}$$

This completes the proof.

The above Collage theorem is the foundation theorem for fractal image coding. It can be directly applied to IFS as follows:

Theorem 2.6 (Collage Theorem for IFS)[Barnsley [3]]

Let $(\mathcal{P},d)$ be a complete metric space. Let $L \in H(X)$ be given, and let $\epsilon \geq 0$ be given. Choose an IFS $\{\mathcal{P}; w_1,w_2,\ldots,w_N\}$ with contractivity factor $0 \leq s < 1$,

so that

$$h(L, \bigcup_{n=1}^{N} w_n(L)) \leq \epsilon$$

where $h(d)$ is the Hausdorf metric. Then

$$h(L, A) \leq \frac{\epsilon}{1-s}$$

where A is the attractor of the IFS.

Equivalently,

$$h(L,A) \leq \frac{h(L, \bigcup_{n=1}^{N} w_n(L))}{1-s} \quad \text{for } \forall\, L \in \mathcal{H}(\mathcal{P})$$

The collage theorem for IFS is crucial for the idea of fractal image coding which throws some light on the construction of IFS codes. Basically, what the theorem says is that if you can take some picture, or portion of a picture, and, by performing affine transformations, end up with smaller (possibly distorted) versions of the original picture which can be placed so that they fill in the original picture without very much overlap and without running very far outside the original picture boundaries and without leaving much of any blank space, then by using these same affine transformations in the deterministic algorithm you can produce a good representation of the original picture.

From the collage theorem we should notice the final attractor is the union of affinely distorted copies of the original, which has global self-similarity. Images which have the property of global self-similarity can be efficiently encoded by IFS.

Theoretically speaking, we can use IFS to encode any black-and-white image. In an extreme case we can transform an image to cover a single point. In the attempt to produce as accurate a collage as possible, there is a second goal that hinders exactness, namely the coding should also be efficient in the sense that as few transformations as possible are used. The definition of an optimal solution to the problem must thus find a compromise between quality of the collage and efficiency. The automatic generation of collages for given target images is a challenging topic of current research.

The collage game is just one example of an entire class of mathematical problems which goes under the category of optimization problems. Such problems are typically very easily stated but are often very difficult to solve even with high-powered, super computer technology and sophisticated mathematical algorithms.

We shall show examples of IFS encoding of two classical shapes: triangle and circle. Given a triangle and connecting the middle points, we get four smaller triangles which

are similar to the original triangle. By choosing the four affine transformations listed in Table 2.4, we can cover the triangle perfectly. Comparing Table 2.4 with Table 2.2, we notice that we just add one extra transformation to that of IFS codes for Sierpinski triangle. Without the extra transformation which covers the middle, we will have another Sierpinski triangle instead of a triangle we want. By Theorem 2.6, these IFS codes should generate an exact triangle.

Our next example is the coding of a circle. A circle can be comprised of nine globally self-similar pieces. Eight transformations cover the edge parts, with one transformation covering the middle part. Table 2.5 are the transformations used to code the circle. The decoded one in Figure 2-8 is nearly a perfect approximation to the original circle.

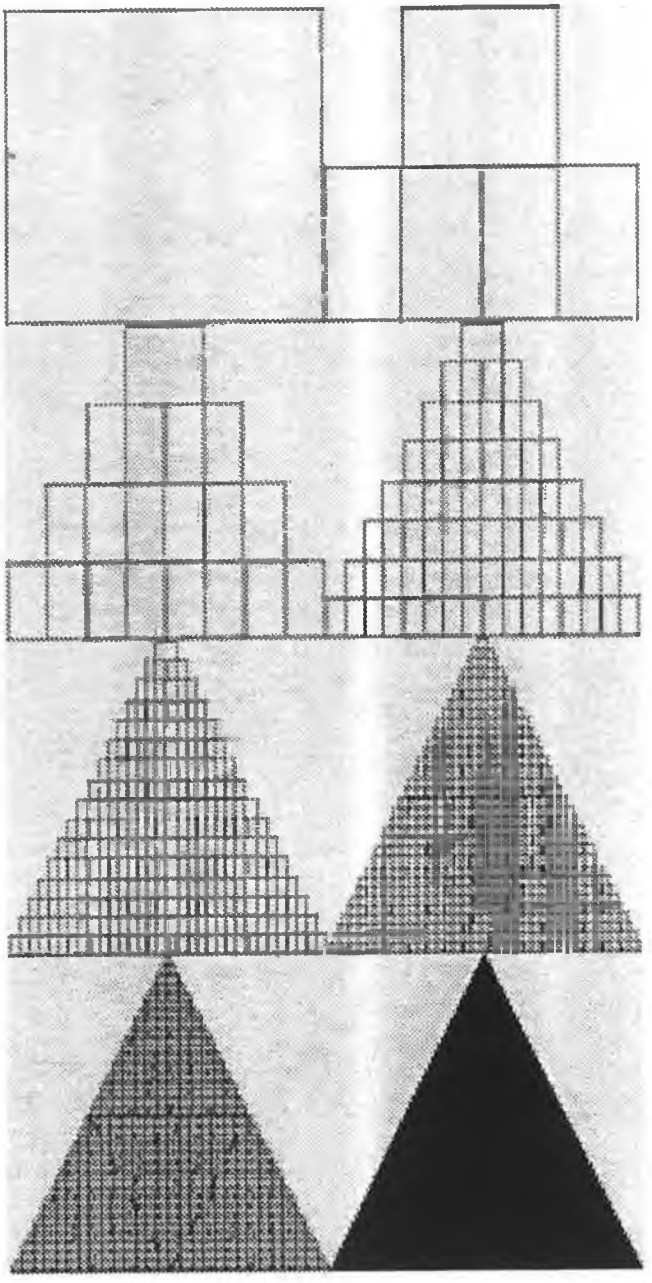


Fig. 2-7 Coding of classical triangle

Table 2.4 IFS codes for a triangle:

w	a	b	c	d	e	f
1	0.5	0	0	0.5	0	0
2	0.5	0	0	0.5	0.5	0
3	0.5	0	0	0.5	0.25	0.5
4	-0.5	0	0	-0.5	0.75	0.5

Table 2.5 IFS codes for a circle:

w	a	b	c	d	e	f
1	.240	-.206	.096	.500	.342	.143
2	.240	-.206	-.096	-.500	.342	.143
3	-.240	.206	.096	.500	-.342	.143
4	-.240	.206	-.096	-.500	-.342	.143
5	.096	.500	.240	-.206	.143	.342
6	-.096	-.500	.240	-.206	-.143	.342
7	.096	.500	-.240	.206	.143	.342
8	-.096	-.500	-.240	.206	-.143	.342
9	.500	0	0	.500	0	0

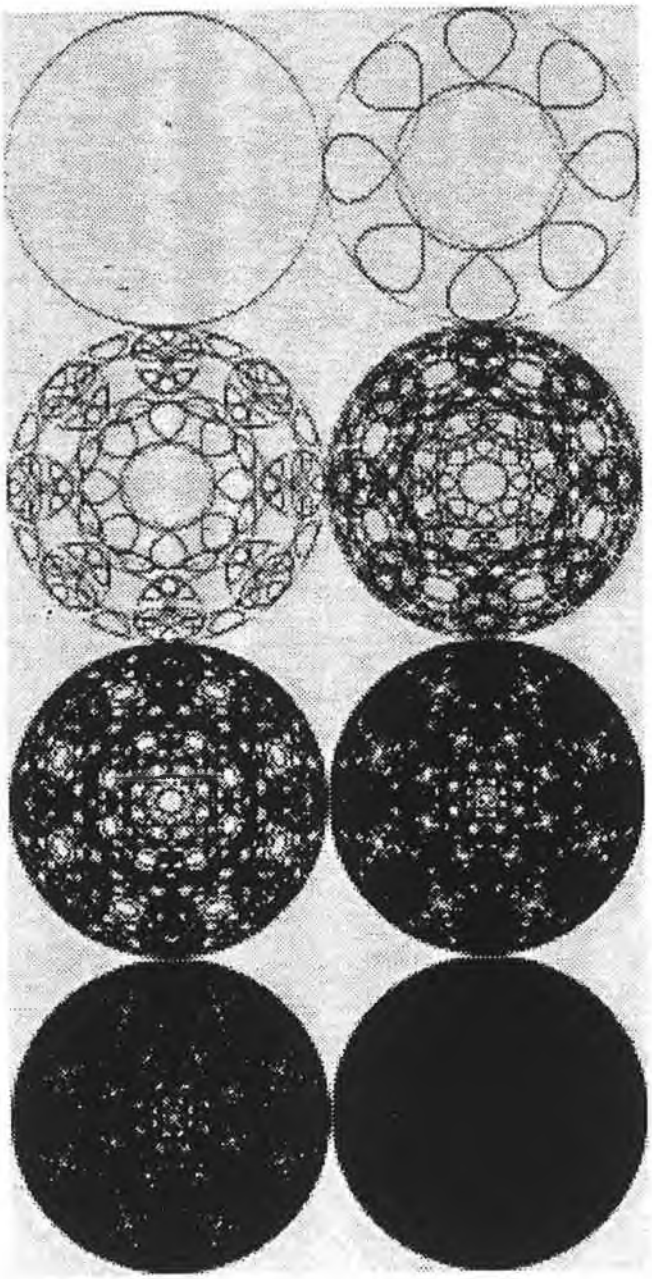


Fig. 2-8 Coding of classical circle

These two examples show us that if images have global self-similarity, it should be possible to find IFS codes for them. Most natural scenes do not possess the property of global self-similarity which makes it very difficult to find the IFS codes. By enlarging the natural images we notice that natural images possess piece-wise self-similarity. The fractal coding algorithm we are interested in is to exploit block-wise self-similarity in natural images. Instead of looking for global self-similarity, we will look for block-wise self-similarity which is a practical way of coding natural images.

2.6 Collage Theorem on Image Space

Let $(\mathcal{P},d)$ denote a metric space of digital images, where d is a given distortion measure, and let p be an original image to be coded. The *inverse* problem of iterated transformation theory resides in constructing an image transformation T defined from the space $(\mathcal{P},d)$ to itself, for which p is an approximate *fixed point*.

Theorem 2.5 (Collage theorem on image space)

Let $(\mathcal{P},d)$ be a complete metric space of digital images, where d is a given distortion measure and let p be an original image to be coded. Let $\tau: \mathcal{P} \rightarrow \mathcal{P}$ be contraction mapping with contractivity factor $0 \leq s < 1$, and let the fixed point of τ be $p_f \in \mathcal{P}$.

Then

$$d(x,x_f) \leq \frac{d(x,\tau(x))}{1-s} \quad \text{for } \forall\, x \in \mathcal{P}$$

The collage theorem on image space states that for a given point p and a given mapping T, the fixed point p_f will be close to the point p if p and Tp are close, and s is not too close to 1, Tp is called the *collage* of p.

The task lies in finding the *contractive* transformation T, which means that there exists an $s \in [0,1)$ such that

$d(T(p), T(q)) \leq s\,d(p,q)$ holds for any pair (p,q) in $\mathcal{P}$.

If the transformation information T has a lower information rate than that of the original image p, image compression can be achieved. In the next chapter we will introduce application of collage theorem to a fractal block coding technique.

Chapter 3.

Fractal Block Coding Technique

3.1 Introduction

Barnsley and Sloan first proposed image encoding by IFS. According to the collage theorem on the image space, image compression can be achieved on condition that we can find a contractive transformation on image space which has a lower information rate that of the original image. However, it is a very challenging undertaking to find such transformations. Under the guidance of Barnsley, Jacquin [9-13] proposed a fractal block coding technique which relies on the assumption that image redundancy can be exploited through *block self-transformability*.

Arbitrary digital greytone images of natural scenes are modelled as fixed points of contractive transformations on the image space. The transformation coefficients are used to iterate successively from any initial image to a fixed point which is close to the original image.

An improved technique in [20] finds the optimized transformation coefficients which improves the coding performance by solving linear equations, thus the coding

efficiency is improved.

In previous implementations of fractal block coding, block-wise self-similarity is used to removein redundancies among images of natural scenes to be coded. A multi-level fractal block coding scheme is proposed in this thesis which exploits affine-block-wise self-similarity at different levels. We extend the block-wise self-similarity to affine-block-wise similarity which significantly enriches the library and improves the coding fidelity. We can achieve a much lower bit rate with good image subjective quality.

In this chapter we also present a very fast coding scheme. The scheme abandons the use of a large number of library blocks. Only the library blocks that cover the target blocks are used. With the introduction of affine-block-wise self-similarity, local image correlations are fully exploited. The simulation results show that the proposed fractal coder is very promising and brings fractal block coding closer to being symmetric.

3.2 Design of Fractal Block Coding System

Fractal block coding is a block based coding method. In this section we will discuss the detailed design and implementation of fractal block coding. A pseudo code of our proposed algorithm is provided.

3.2.1 Image Partition Strategy

Given an image p of natural scenes with size of $N \times N$, we first partition it into non-overlapping square blocks of two different sizes, thus forming a *two-level square partition*. Smaller size BxB blocks are referred as *target blocks* from p which are to be coded. Larger size DxD blocks are referred as *library blocks* also from p which are used to search through when coding each target block. D:B is empirically chosen as 2:1.

We seek to exploit similarities between each target block and library blocks under Jacquin's assumption that images are block-wise self-similar.

3.2.1.1 Multi-Level Target Blocks

According to the image to be coded, a target block size is chosen to trade image quality against compression ratio in previous implementations of fractal block coding. Usually a target block size of 8x8 is chosen in a high resolution image and 4x4 is chosen in coding a low resolution image. The bigger the target block size, the more fine details are lost. Because of the fixing of the target block size, self-similarity at higher levels is not exploited. Images of natural scenes should have self-similarity at different scales. We propose a multi-level scheme which always attempts to exploit

self-similarity at the highest possible level. If a block fails to be coded at a higher level, it will be coded at a lower level. By exploiting image self-similarity at different levels, we can achieve a higher compression with good subjective image quality.

In the multi-level fractal block coding scheme, we first set the minimum target block as $min\_size \times min\_size$, and the maximum target block as $2^N min\_size \times 2^N min\_size$. The n^{th} level of target blocks consists of target blocks whose size is $2^n min\_size \times 2^n min\_size$. Thus we have coding levels ranging over 0,1,2,...,N-1,N. For example, target block sizes of 4×4, 8×8, 16×16, 32×32, correspond to coding levels 0 up to 3, where the minimum target block size is set to 4×4.

An error threshold is set for each higher level except the lowest level corresponding to the smallest target block size. The biggest and the smallest target block size, together with all the error thresholds, are free parameters depending on the nature of the application and quality desired. Coding is implemented, starting from the highest level which has the largest target block size, down towards the lowest level with the smallest target block size. Hence, coding is realized in a way that smoother areas are first coded with larger blocks while areas with finer details are coded with smaller block size at the lower levels. Therefore, a lower bit rate could be achieved with good subjective quality.

3.2.1.2 Generation of Library Blocks

In previous implementations of fractal block coding, library blocks are generated from a uniform non-overlapping partition. This is equivalent to moving a window of library block size vertically and horizontally across the whole image at the step of target block size or library block size. If the target block size is chosen as 4×4, then we have library blocks of size 8×8. After the partition, we intend to exploit block-wise self-similarity between target blocks and library blocks.

Fractal block coding is a lossy scheme. The coding fidelity heavily depends on the finding of a library block which could be transformed to cover the target block well. However, in our simulation we find the scheme frequently fails to find a library block which can be transformed to cover the target block well enough after an exhaustive search through all the library blocks. Image parts with fine details, especially sharp edges, are difficult to code resulting in annoying "staircase" effects.

To overcome this limitation, we extend block-wise self-similarity to *affine-block-wise* self-similarity. Image redundancies are removed by exploiting affine-block-wise self-similarity. The library blocks are not required to be square blocks, but they can be affinely distorted square blocks. With our proposed scheme, a much richer library is obtained. Therefore, the probability of finding good matches is significantly increased. Images coded by the proposed scheme can have better subjective quality.

Instead of moving all of these different shaped windows across the whole image for the generation of these additional library blocks, we create these shapes of library blocks at the time of coding. They are simply the affine transformed shape of the square library blocks after the initial blockwise partition. The information needed about this new library block is the address of the square library block and the index of its associated affine transformation.

After creating the initial non-overlapping square blocks as library blocks, each library block undergoes several designated affine transformations which includes rotations in x direction, y direction or both directions. Thus the similarity of searching block-wise self-similarity is extended to affine-block-wise self-similarity.

3.2.1.3 Choice of Affine Shapes

By taking into account different orientations of square library blocks, the library is substantially enriched, thus increasing the chance of finding a library block with the best match at a higher level. However, because of the introduction of more shapes of library blocks, extra bits are needed to describe these shapes. According to Table 3.2, extra bits are needed to describe the affine transformation of the square library block we introduce. However, the cost of few more bits can be offset by the gain that more

blocks are coded at higher target block levels.

With the introduction of affine-shape library blocks, more searching effort is needed for each target block. We can have the option of not using these different shaped library blocks if a satisfactory match is found. Another possible strategy is that we use them for fine tuning after the best block-wise self-similarity is found instead of using all the affine library blocks. Thus the increase in computation is not as significant.

To improve coding efficiency, we need to further investigate the frequency of the occurrence of each affine-shapes. We first try out all the combinations of rotations in x and y. We have carried simulations to cover most combinations of x and y and we choose rotation angles in x and y directions as follows:

x directions { 67.5 45.0 22.5 0 -22.5 -45.0 -67.5 }

y directions { 67.5 45.0 22.5 0 -22.5 -45.0 -67.5 }

By having all the possible 49 combinations, we can see the frequency of the usage of these affine shapes.

Table 3.1 Frequence of the usage of 49 affine shapes(percent %)

a1: 22.5 degrees a2: 45.0 degrees a3: 67.5 degrees							
	a3	a2	a1	0	-a1	-a2	-a3
a3	4	2	2	3	1	0	0
a2	1	1	1	3	2	1	0
a1	1	1	1	1	4	2	1
0	3	2	3	5	2	1	1
-a1	1	2	5	5	1	1	1
-a2	3	1	2	3	1	1	2
-a3	6	3	2	4	2	1	2

From above Table 3.1, we notice that the usage frequencies of the combinations are quite diferent from each other. Some of them are more frequently used than the others. We then carried some simulations to show the relationships of number of shapes, coding efficiency, coding quality and entropy.

Table 3.2 Relations of frequency, entropy and PSNR

Usage Frequency from Table(3.1)		Top 6	≥4%	≥2%	ALL
Number of Affine Shape Used	0	6	13	23	49
Entropy	1.0	2.35	3.24	3.82	4.54
PSNR(dB)	31.94	32.41	32.62	32.76	32.79

Note: throughout this thesis, all the entropies reported are calculated according to the formula in 1.2 by using the frequencies of occurrence.

In Table 3.2, the first column of data represents the case of affine shapes not used. We can see that the more affine-shapes introduced, the better the image fidelity at the cost of more bits and more computational effort. By listing the 49 affine shpaes in decending frequency order, we can trade coding efficiency against fidelity by using different number of affine-shapes at the top of the list.

All our simuations in the following will use the six affine-shapes which are used most freqently. The top six chosen are:

(-67.5, 67.5), (-22.5,22.5), (-22.5,0),

(22.5 ,-22.5), (-67.5 ,0.0), (67.5,7.5).

3.2.2 Construction of Transformation Codes

To exploit the similarities among *target blocks* and *library blocks*, we allow the following transformations on a library block:

1) Affine transform *library blocks* to the shape of a square block:

$$d \rightarrow A(d)$$

2) Scale to the size of the *target block*:

$$d \rightarrow S(A(d))$$

3) $S(d)$ is isometrically transformed during which pixels are shuffled without changing the norm.

$$S(A(d)) \rightarrow G(S(A(d)))$$

4). Greytone is scaled by a real factor:

$$G(S(A(d))) \rightarrow \alpha \cdot G(S(A(d)))$$

5). and added to a simple greytone translation:

$$\alpha \cdot G(S(A(d))) \rightarrow \alpha \cdot G(S(A(d))) + t$$

These transformations of a library block are also described in Figure 3-1.

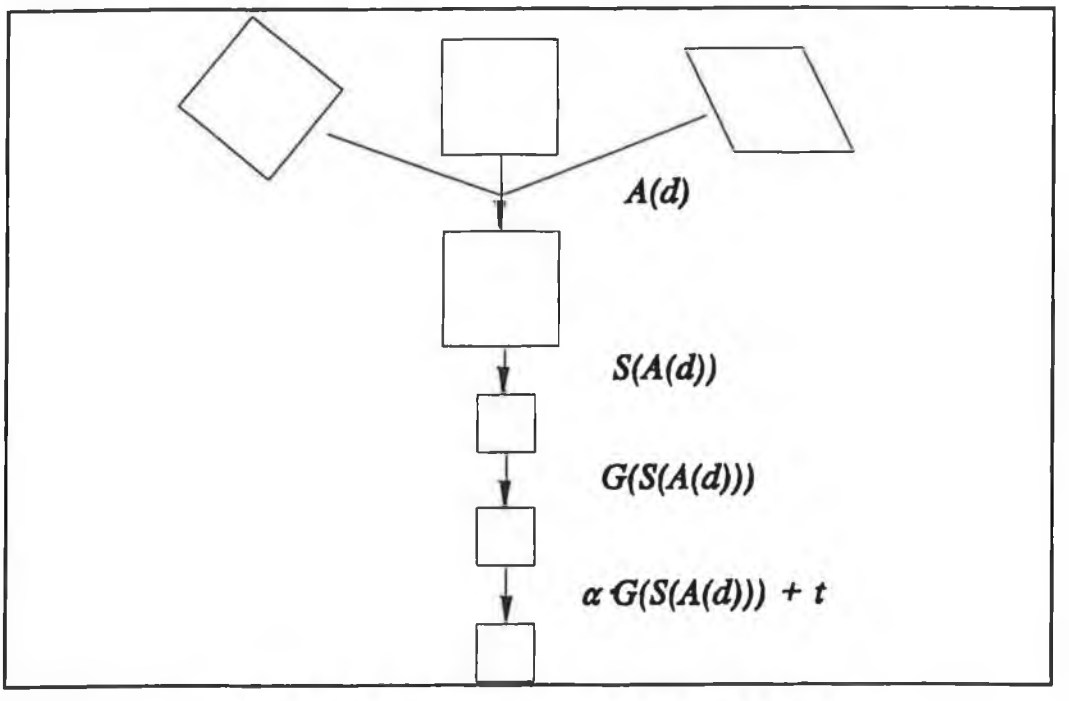


Fig. 3-1 Transformation of a library block

Isometric transformations which simply shuffle pixels within a library block are called isometries. The following is a list of eight canonical isometries of a square block s with size n×n.

1) Identity:

$$(I_0 s)_{i,j} = s_{i,j}$$

2) Orthogonal reflection about the mid-vertical axis ($j = (B\text{-}1)/2$) of the block:

$$(I_1 s)_{i,j} = s_{i,B-1-j}$$

3) Orthogonal reflection about the mid-horizontal axis ($i = (B\text{-}1)/2$) of the block:

$$(I_2 s)_{i,j} = s_{B-1-i,j}$$

4) Orthogonal reflection about the first diagonal ($i = j$) of the block:

$$(I_3 s)_{i,j} = s_{j,i}$$

5) Orthogonal reflection about the second diagonal ($i + j = B - 1$) of the block:

$$(I_4 s)_{i,j} = s_{B-1-j,B-1-i}$$

6) Rotation around the centre of the block, through + 90 degrees:

$$(I_5 s)_{i,j} = s_{j,B-1-i}$$

7) Rotation around the centre of the block, through +180 degrees:

$$(I_6 s)_{i,j} = s_{B-1-i,B-1-j}$$

8) Rotation around the centre of the block, through -90 degrees:

$$(I_7 s)_{i,j} = s_{B-1-i,j}$$

Although these eight isometries are unique, the usage frequencies are significantly different. We will discuss this further in 3.6.

These transformations can be used to define a transformation T: $\mathcal{P} \rightarrow \mathcal{P}$. By associating with each target block p_{ij} in p a transformation such that the (k,l) block in T_{ij} is given by

$$[T_p]_{kl} = \begin{cases} \alpha \cdot G_{ij}(S(d^{(i,j)}) + t_{ij} & \text{if } (i,j) = (k,l) \\ 0 & \text{otherwise} \end{cases}$$

we can define T in a blockwise manner,

$$T = \sum_{k=1}^{n} \sum_{l=1}^{n} T_{kl}$$

and obtain

$$[T_p]_{kl} = [T_{kl}p]_{kl} \quad \text{for all } k,l \in \{1,\ldots,n\}$$

The coding of each target block (k,l) is to choose the library and isometry, and to compute α and t, such that the distance $d(p,Tp)$ is minimized.

From above scheme, we can notice that each target block is to be covered by a transformed library block which is a good approximation. If we have target blocks overlapped, the overlapped parts will be covered by multiple transformed library blocks. As long as each target block is well covered by a transformed library block, the overall transformation on the original image will be a good approximation to the original image. The overlapped parts will have multiple values because they are covered by multiplue transformations. However, this will not affect the decoding procedure. The reason is that only the last transformation dominants the value of an overlapped part. Values of the rest of transformations will be overwritten by the last

dominant transformation, therefore we will have a unique decoded image.

Simulations were carried out 352x288 "Miss America" at level 8x8 to compare non-overlapping with overlapping partitions of target blocks. We achieve a bit rate of 0.30 bpp at a PSNR of 34.15 dB with 1584 non-overlapping target block partition. We have a bit rate of 0.86 bpp at a PSNR of 34.15 dB with 6177 overlapping target blocks. We can notice the obvious efficiency advantage of non-overlapping partion. Those overlapped parts are repeatedly coded which costs extra bitrate and time. This is the reason why we should chose all the target blocks non-overlapped. By reducing the number of target blocks, we can achieve lower bitrate and better efficiency.

3.2.3 Distortion Measure

Each target block is a real $B \times B$ matrix; i.e. an element in $\mathfrak{B} = R^{B \times B}$. We can introduce a metric on $\mathfrak{B}$ by defining an inner product $\langle \cdot, \cdot \rangle$: $\mathfrak{B} \times \mathfrak{B}$:

$$\langle x,y \rangle = \sum_{m=1}^{B} \sum_{n=1}^{B} x_{mn} y_{mn} \quad \textit{for all } x,y \in \mathfrak{B}.$$

This yields the r.m.s. metric

$$d_B(x,y) = \langle x-y, x-y\rangle^{1/2} = |x-y| = \left(\sum_{m=1}^{B}\sum_{n=1}^{B}(x_{mn}-y_{mn})^2\right)^{1/2}$$

on $\mathfrak{B}$. Now, we can choose a metric on $\mathcal{P} = R^{nB\times nB}$ in many ways, for example

$$d_1(p,q) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_B(p_{ij},q_{ij}),$$

or

$$d_2(p,q) = \max_{i,j} d_B(p_{ij},q_{ij}).$$

These metrics have in common that they are minimized if $d_B(p_{ij},q_{ij})$ is minimized independently for all (i,j). We therefore reduce our task to the following:

For each target block p_{ij} minimize

$$d_B(p_{ij},[Tp]_{ij}) = d_B(p_{ij}, \alpha \cdot G(S(d))+t)$$

3.2.4 Calculation of α and t

Then, for each combination of library block d and isometry G, we are trying to approximate a given target block b with a block $b' = \alpha c + t$ where $c = G(S(d))$. In the original work of Jacquin [9-13] t was chosen as a block constant greytone; $t = r \cdot m$ where m is a constant greytone block and r is some real number. Optimal coefficients α and t were found by using the projection theorem in [20]. b' can be regarded as an element of a two-dimensional linear subspace spanned by the basis $\{s_1,s_2\} = \{m,c\}$.

By the projection theorem [28], the optimal coefficients α and r that minimize $d_B(b,b')$ are found by letting $b - b'$ be orthogonal to this subspace:

$$\langle b-b', s_i \rangle = 0; \quad i = 1,2.$$

From the above orthogonality equations we have:

$$\alpha \langle c,c \rangle + r \langle m,c \rangle = \langle b,c \rangle$$

$$\alpha \langle c,m \rangle + r \langle m,n \rangle = \langle b,m \rangle$$

By letting

$$a_{11} = \langle c,c \rangle \quad a_{12} = \langle m,c \rangle$$

$$a_{21} = \langle c,m \rangle \quad a_{22} = \langle m,m \rangle$$

$$b_1 = \langle b,c \rangle \quad b_2 = \langle b,m \rangle$$

$$A = (a_{ij})_{2\times 2}$$

we have

$$A \begin{pmatrix} \alpha \\ r \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

If $det(A) = a_{11}a_{22} - a_{12}a_{21} \neq 0$ then A^{-1} exists, we have a unique solution

$$\begin{pmatrix} \alpha \\ r \end{pmatrix} = A^{-1} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$where \quad A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$$

By this algorithm we can get the optimized coefficients for α and r directly by the above simple calculations.

3.3 Algorithm for Multi-Level Fractal Block Coding

The algorithm of our proposed multi-level fractal block coding is described as follows:

parameter initialisation:
biggest target block size: $2^N min\_size \times 2^N min\_size$
smallest target block size: $min\_size \times min\_size$
define error threshold level $error\_threshold_l$
corresponding to level l

for coding level l from N to 0{
  for each target block p_{ij}{
    for each library block: d_{mn}{
      affine transform d_{mn} *to* $S(d_{mn})$
      for each isometry k{
        calculate α *and r*
        $p' = \alpha S(d_{mn}) + r$
        calculate $d_B(p_{ij}p')$ *(distance between* p_{ij} *and p')*
        keep the codes with smallest $d_B(p_{ij}p')$
      } /*end of k*/
    } /*end of m,n*/
    if ($minimum\ d_B(p_{ij}p') <$ *corresponding threshold or N == 0)*
    output codes of current target block
  } /*end of i,j*/
  new target blocks p_{ij} *are formed by splitting up*
  uncoded target blocks.
  new library blocks d_{mn} *are created from the uncoded target blocks.*
}/*end of l*/

Fractal codes for each target block are target address, library address, isometry number, α and t. Quantization of the codes of the target block is discussed in the following section.

3.4 Searching Strategy

During the coding of a target block, we can use the library blocks from the whole image which is computationally costly. In previous implementations of fractal block coding, library blocks from the same quadrant image are exhaustively searched[9-13]. In our scheme, coding of a target block could be accomplished as soon as a good match is found instead of searching the whole quadrant. Also we define a disk area of library blocks centered at the target block. This is due to the fact that self-similarity most probably happens in the neighbourhood area. In natural scenes, nearby texture could be somewhat similar or quite similar. A block on the face should be best matched by a nearby library block target which is most probably a block on the face area.

The search radius should be big enough to provide good reconstructed image quality. The bigger the searching radius, the better the probability of finding a good library block at the cost of more computational complexity. The smaller the search radius, the worse the image quality we have.

3.5 Error Threshold and Image Fidelity

At each higher coding level, an error threshold is set. We can control these error

thresholds to trade image quality against bit rate. Obviously the higher the error threshold, the more blocks will be coded at the corresponding higher level at the cost of lower image quality. We have carried out simulations on the choice of thresholds. To have reconstructed images with acceptable image quality, we carried out extensive simulations trying out the effect of error thresholds on image subjective quality.

We first set the threshold 3 for each level which gives an excellent image fidelity. By increasing the threshold of one level while keeping other thresholds fixed, we notice the degradation of image fidelity from Fig.2-3.

Although PSNR value decreases the slowest at level 32 × 32, it does not mean the image quality degenerates the least. We need to have subjective tests to see how the threshold affects the subjective image quality.

Subjective tests carried out (by four Ph.D research students in video coding group of DCU) suggests the choice of thresholds for each coding level which are listed in Table 3.3.

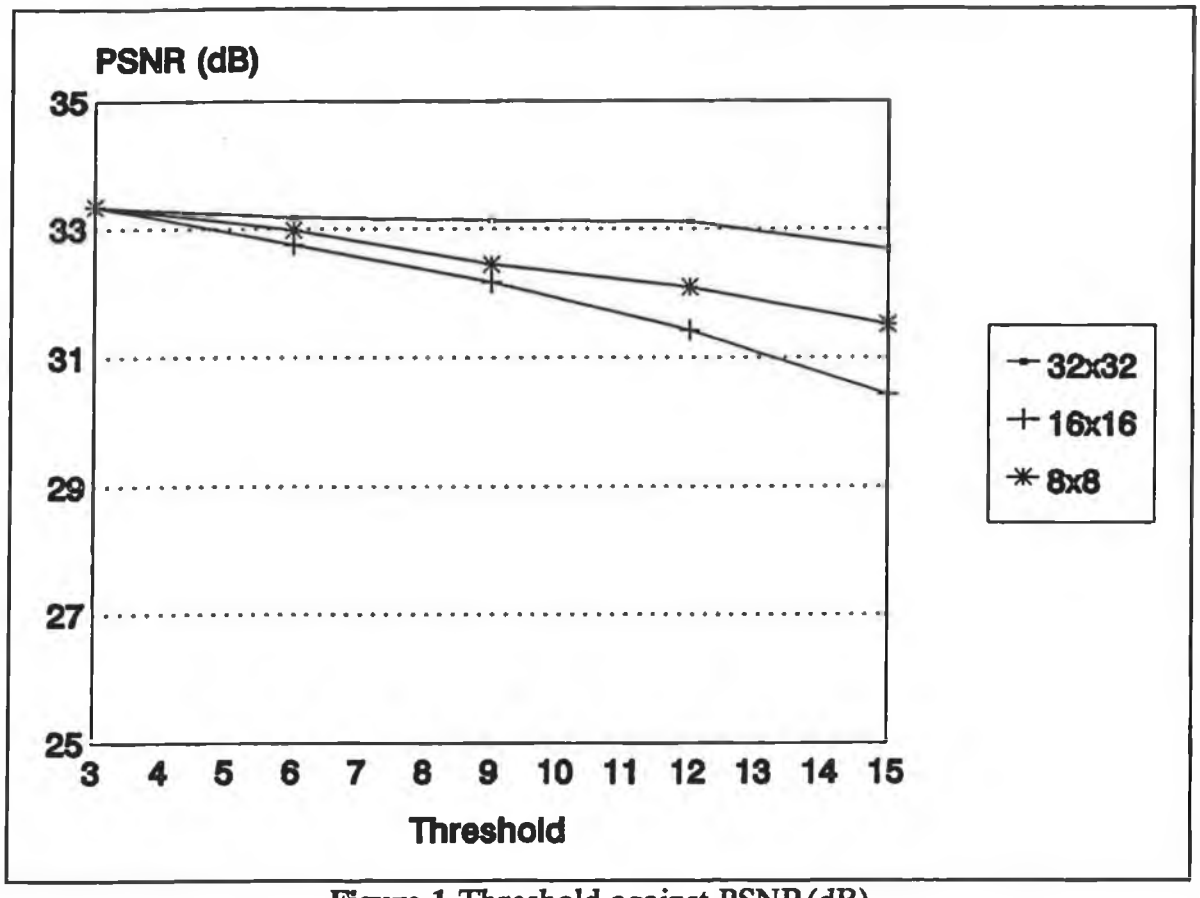


Figure 1 Threshold against PSNR(dB)

Figure 3-2 Threshold vs. PSNR(dB)

Table 3.3 Threshold ranges of different coding levels

LEVEL	TARGET SIZE	THRESHOLD
3	32 × 32	<= 5
2	16 × 16	<= 5
1	8 × 8	<= 7

The above table provides us with a choice of threshold. To have good subjective image quality, thresholds of each level should be chosen within their corresponding limits. We can sacrifice image fidelity to achieve lower bit rate by using higher thresholds.

3.6 Choice of Isometries

We have mentioned in 3.2.2 that the usage frequency of eight isometries should be studied to improve coding efficiency. By studying the usage frequency of each isometry, we can improve coding efficiency.

Results of eight simulations regarding to isometry usage frequency and PSNR are reported in Table 3.4 where we can see that isometries 6 and 7 have much lower frequency than other isometries. The most frequently used one is the original one. By removing the isometries 6 and 7, the degradation of the image is negligible.

Table 3.4 Simulation results of usage frequency of isometries

ISO / FREQ / SIM %	1	2	3	4	5	6	7	8	PSNR (dB)
1	30	8	15	10	11	4	5	12	33.15
2	31	9	16	11	12	0	6	12	33.12
3	31	10	16	11	12	4	0	12	33.11
4	32	0	15	11	12	5	8	13	33.10
5	34	9	16	0	12	0	6	14	33.09
6	31	11	17	12	13	0	0	13	33.08
7	35	0	18	14	15	0	0	15	32.97
8	41	0	20	17	0	0	0	19	32.80

To improve coding efficiency, the isometries should be arranged in such a way that their usage frequencies is in decending order. Therefore frequently used isometries are first tried to avoid fruitless effort on least frequently used ones.

3.7 Quantization of Transformation Codes

Fractal codes for each target block are target block address, library block address and

its associated affine transformation, isometry number, α and t. We will discuss the quantization of these codes.

The address of target block and library block includes the x and y coordinates which are the multiples of corresponding block size. In order to reduce the bits to represent these x and y coordinates, we propose an output of codes in the following way:

The output of the fractal codes is in a raster order of the biggest target block size level. Each biggest one could possibly be coded at combinations of lower level target block size. If it is not coded at the highest level, it must be split into four smaller pieces, say A, B, C, and D. The codes for the whole small piece must be sent before sending codes for the next small piece. i.e. after all the codes covering piece A are sent, we start sending codes for B. If A, B, C and D are coded at lower levels, it is done in the same manner as A,B, C and D. This is illustrated by an example in Figure 3-3. In Figure 3-3, a 32×32 block is coded by different target block levels.

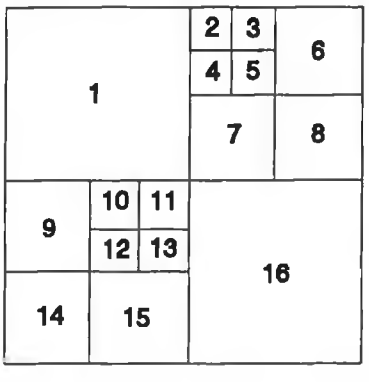


Fig. 3-3 A 32×32 block coded at different levels

If we use characters *A,B,C,D* to represent fractal codes for corresponding 32×32, 16×16, 8×8 and 4×4, the output of codes is as follows according to our proposed flow:

B 1 , *D* 2 3 4 5, *C* 6, *C* 7, *C* 8, *C* 9, *D* 10 11 12 13, *C* 14, *C* 15, *B* 16

Entropy of transform codes at different levels is listed in Table 3.5.

Table 3.5 Entropy table for slow scheme using searching strategy in 3.4

target size	x_t & y_t	x_l	y_l	t_{xy}	k	α	r	TOTAL (bits)
32 × 32	1.0	2.13	2.42	1.90	2.28	4.23	3.17	17.13
16 × 16	1.0	3.61	3.66	2.39	2.52	5.57	5.46	24.21
8 × 8	1.0	3.86	3.92	2.54	2.61	5.90	6.01	25.84
4 × 4	0.25	3.90	3.70	2.58	2.78	6.09	5.93	25.23

3.8 A Fast Coding Scheme

The fractal block coding scheme is asymmetric. Coding is much more computationally costly than decoding, since encoding a target block involves a large amount of calculation in searching through library blocks to find a library block which is a good

match to the target block. As stated in section 3.4, library blocks in the neighbourhood area are used. Beaumont[14] shows that most of the library blocks are found in the neighbourhood of a target block. However, our simulations show that using library blocks only in the very close neighbourhood, the subjective quality of the image is quite poor. The radius of the neighbourhood should be big enough to ensure we have enough library blocks to search through to find a good match.

The most direct way to reduce coding complexity is to reduce the search effort. If we can reconstruct an image coded by using just a few libary blocks, the coder can be hundreds times faster. In our simulations, we see that the subjective image quality is very poor in the reconstructed image coded using the library blocks in the very close neighbourhood, however, parts of the image quality are acceptable. As for the coding efficiency, this is very attractive to us, provided that the reconstructed image has acceptable image quality. This is the reason for our interest in coding the target block by exploiting self-similarity in the very close neighbourhood of the target block to be coded to improve coding efficiency.

We introduce using the library blocks that contain target library blocks at one of its corners. Therefore target blocks at the four corners have only one library block. Target blocks in the middle of the image have four library blocks and target blocks at the edges have two library blocks. Therefore the number of library blocks ranges from 1 to 4. If a target block is not block-wise self-similar to the library blocks it has, affine-

block-wise self-similarity is exploited.

Simulations have shown that local correlations can be removed by the exploitation of block-wise self-similarity together with affine-block-wise self-similarity in the very local area of the target block. The very local library blocks are sufficient to provide a good match. The subjective image quality of the reconstructed image by this scheme is quite good.

Apart from the tremendous reduction in coding complexity, the bits used to express the library block address are reduced down to 2 bits which brings down the bit rate. This scheme is a very promising one and brings fractal block coding closer to real-time application.

Entropy of transform codes at different levels is listed in Table 3.6.

Table 3.6 Entropy table for fast scheme

target size	x_t y_t	x_l y_l	t_{xy}	k	α	r	TOTAL (bits)
32 × 32	1.0	1.97	1.90	2.28	4.23	3.17	14.55
16 × 16	1.0	2.00	2.39	2.52	5.57	5.46	18.94
8 × 8	1.0	2.00	2.54	2.61	5.90	6.01	20.06
4 × 4	0.25	1.98	2.58	2.78	6.09	5.93	19.61

3.9 Fractal Decoding

The fixed point of the transformation T is the limit of any sequence of iterates. Therefore transformation codes can be used to iterate successively on any initial image which will approach the fixed point which is the approximation of the original image we coded upon. The iteration procedure is interrupted when two consecutive iterates are very close. Through simulations we have found that after five iterations, signal-to-noise ratio ceases to increase. Thus a fixed iteration time of five is usually given which seems to be sufficient. This is due to the strong contractivity of the mapping T. The decoded image is therefore

$$p_{decoded} = T^5(0),$$

where 0 denotes an image with a greytone value of zero in every pixel.

3.10 Simulation Results

Simulations one to three are carried out on "Miss America" with a size of 256×256 at 8 bpp. Figure 3-4 is the original image of "Miss America". These reconstructed images are coded by using a two-level partition which shows the relationship between target blocks size, bit rate and image quality.

Simulation four and five are carried out on the ISO image "Lena" with a size of

512×480 at 8 bpp. Figure 3-12 is the original of "Lena". At low bit rates, we can see our first scheme with affine-block-wise self-similarity gives better subjective quality than block-wise self-similarity. With our proposed scheme, we can achieve good image subjective quality at much lower bit rate.

In simulation five, we first show the reconstructed "Lena" using library blocks in a radius of 4 and 8 times of target block size. Simulation results of our fast coding scheme are described in section 3.8.

3.10.1 Simulation One

We first use target block size 4×4 to code "Miss America". Figure 3-5 is the reconstructed image. We notice that the reconstructed image is of very good subjective quality at the cost of high bit rate. All the fine details are very well preserved. Target block size 4×4 is the smallest size we can use to achieve a compression. Target block size 3×3 is not used because it gives very little gain in compression. Target block size 2×2 could result in an increase of bits to express the original image instead of compression.

3.10.2 Simulation Two

We then code "Miss America" with a target block level of 8×8. The reconstructed image(Figure 3-6) becomes quite fuzzy and fidelity around the eye areas is lost. Serious blocking artifacts can be noticed. Target size of 8×8 seems too big for coding image areas with fine details on this image. However we notice that smoother areas seem to have better image quality.

At a target block size of 8×8, if we use affine-block-wise self-similarity in addition to square block-wise self-similarity we will have better subjective image quality as in the reconstructed image Figure 3-7. This shows that fractal block coding with the exploitation of affine-block-wise similarity does increase the chance of finding a good match of self-similarity. Image fidelity is better preserved by the exploitation of affine-block-wise self-similarity at a low bit rate.

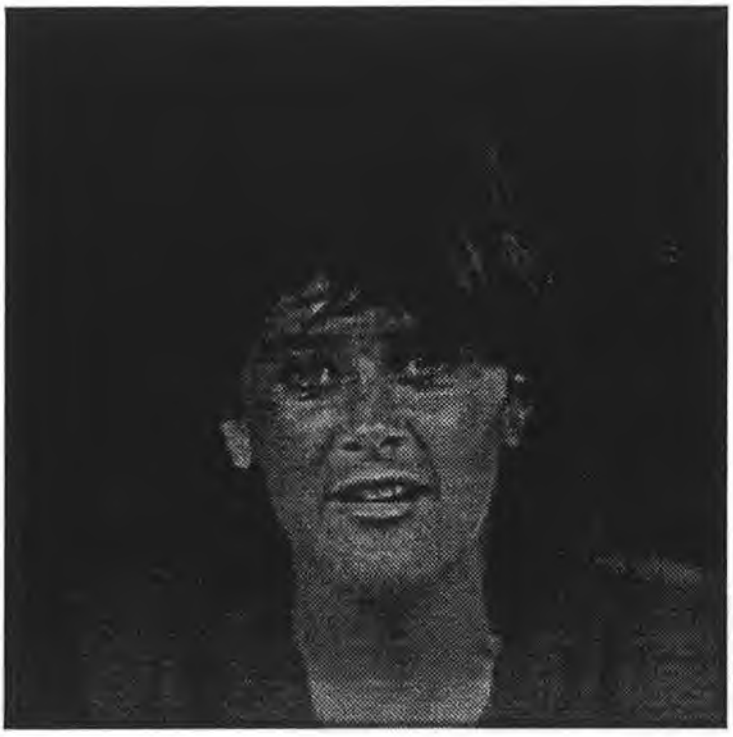


Fig. 3-4 Original image "Miss America" with 256 × 256 at 8 bpp

Fig.3-5 Reconstructed image of "Miss America" coding level 4×4 with pure block-wise self-similarity (1.75 bpp at 36.7 dB)

Fig.3-6 Reconstructed image of "Miss America" coding level 8×8 with pure block-wise self-similarity(0.41 bpp at 32.7 dB)

Fig.3-7 Reconstructed image of "Miss America" coding level 8×8 with affine-block-wise self-similarity(0.47 bpp at 33.36 dB)

3.10.3 Simulation Three

In the implementation of our proposed multi-level fractal block coding scheme, the choice of biggest target block size and smallest target block size together with the thresholds are decided according the application and image quality desired. Figure 3-8 is the reconstructed image with its corresponding coding parameters listed in Table 3.7.

Decoding is always the most fascinating aspect of fractal image coding. Figure 3-10 is the decoding procedure starting with a uniform black image. Figure 3-11 is the decoding procedure starting with "Lena" image. However, the final fixed points which is our reconstructed image are about the same. Once you have the fractal codes, you can reconstruct the coded image starting from any image. Figure 3-10 shows the first four iterations starting from a uniform greylevel image. The first iteration is shown from the highest level to the lowest level which tells that this image was coded from smooth areas towards fine detail areas.

In order to achieve a lower bit rate, we have to code as many blocks at higher target block level as possible. By choosing bigger thresholds at each level we can have a higher compression ratio. The reconstructed image(Figure 3-9) is coded with the choice of new thresholds in Table 3.8.

Fig.3-8 Reconstructed image of "Miss America" coded from level 32 × 32 down to 4 × 4 (Table 3.7)

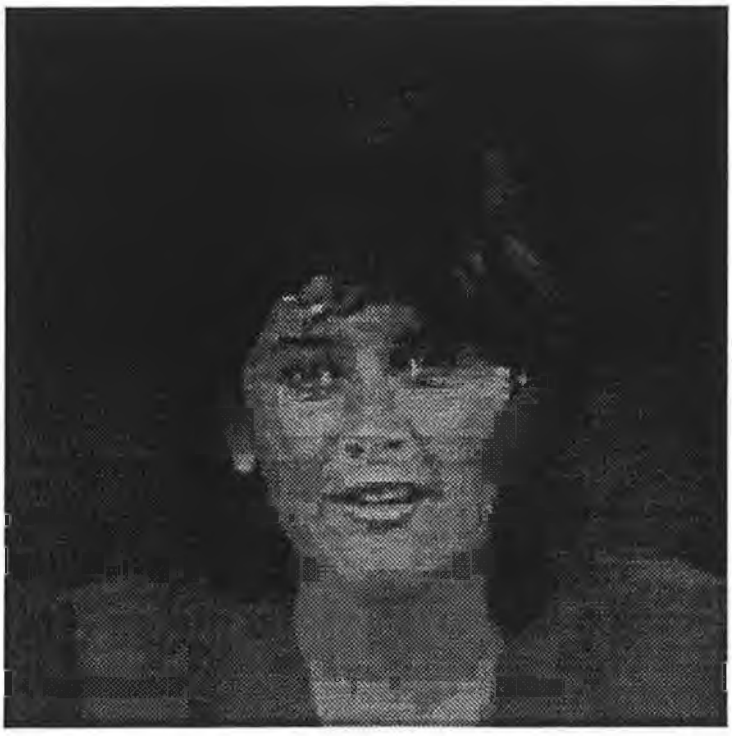


Fig.3-9 Reconstructed image of "Miss America" coded from level 32×32 down to 4×4 (Table 3.8)

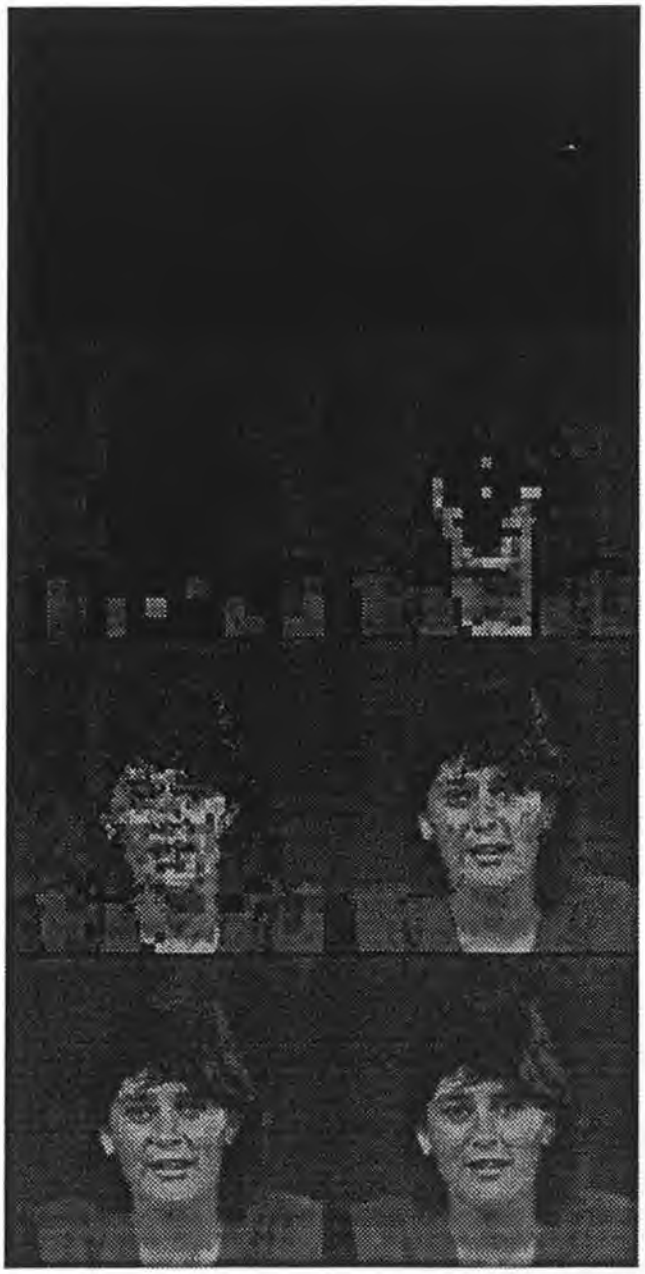


Fig.3-10 Decoding starting from a uniform image(gl = 0)

Fig.3-11 Decoding starting from a "Lena"

3.10.4 Simulation Four

In a previous implementation [20] "Lena" was coded at 8×8 target block size level. The bit rate achieved was 0.5 bpp with PSNR of 30.8 dB. With our muti-level fractal block coding scheme, a much lower bit rate with good image quality can be achieved.

Figure 3-13 is the reconstructed image using our slow scheme where only block-wise self-similarity is exploited. Corresponding coding specifications are listed in Table 3.9. The sharp edge of the hat is not well preserved; a "stair-case" effect can be noticed. The reason for this is at target block size 8×8 some fine detail areas like sharp edges fail to be matched by a good library block after an exhaustive search through all the library blocks.

By using affine-block-wise self-similarity we have a reconstructed image as shown in Figure 3-14 with improved subjective image quality. This demonstrates that the extension of block-wise to affine-block-wise does increase the chance of coding a target block at higher levels.

Figure 3-15 is the reconstructed image using our scheme with affine-block-wise self-similarity which gives much better subjective quality. The coding specification is listed in Table 3.11. With a choice of threshold of 5,5,7 (Table 3-12), a bit rate of 0.52 bpp is achieved with a PSNR of 34.34 dB.

Fig.3-12 Original "Lena" 512×480 with 8 bpp

Fig.3-13 Coded from level 32×32 downto 8×8 by exploiting pure block-wise self-similarity (Table 3.9)

Fig.3-14 Coded from level 32×32 downto 8×8 by exploiting affine-block-wise self-similarity (Table 3.10)

Fig.3-15 Coded from level 32×32 downto 8×8 by exploiting affine-block-wise self-similarity (Table 3.11)

Note: Throughout this dissertation, PSNR we use is the peak signal-to-noise ratio [Appendix A].

Table 3.7 Coding specification for "Miss America" with pure block-wise

Image Coded:"Miss America" Size: 256×256 8 bpp				
LEVEL	Target size	Threshold	Entropy	Blocks coded
3	32×32	3	17.13	17
2	16×16	6	24.21	104
1	8×8	9	25.84	241
0	4×4		25.23	380
0.28 bpp PSNR: 34.9 dB Fig.3-8				

Table 3.8 Coding specification two for "Miss America" with affine-shapes

Image Coded:"Miss America" Size: 256×256 8 bpp				
LEVEL	Target size	Threshold	Entropy	Blocks coded
3	32×32	6	17.13	22
2	16×16	9	24.21	101
1	8×8	12	25.84	204
0	4×4		25.23	256
0.22 bpp PSNR: 34.1 dB Fig.3-9				

Table 3.9 "Lena" coded by exloiting purge block-wise self-similarity

Image Coded:"Lena" Size: 512×480 8 bpp				
LEVEL	Target size	Threshold	Entropy	Blocks coded
2	32×32	3	17.13	27
1	16×16	6	24.21	401
0	8×8		25.84	1804
0.23 bpp PSNR: 29.5 dB				

Table 3.10 "Lena" coded by slow scheme with affine-wise self-similarity

Image Coded:"Lena" Size: 512×480 8 bpp				
LEVEL	Target size	Threshold	Entropy	Blocks coded
2	32×32	6	17.13	29
1	16×16	9	24.21	428
0	8×8		25.84	1664
0.22 bpp PSNR: 30.1 dB Fig. 3-14				

Table 3.11 "Lena" coded by slow scheme with affine-wise self-similarity

Image Coded:"Lena" Size: 512×480 8 bpp				
LEVEL	Target size	Threshold	Entropy	Blocks coded
2	32×32	6	17.13	59
1	16×16	12	24.21	419
0	8×8		25.84	1220
0.17 bpp SNR: 29.7 dB				

Table 3.12 "Lena" coded by slow scheme with affine-wise self-similarity

Image Coded:"Lena" Size: 512×480 8 bpp				
LEVEL	Target size	Threshold	Entropy	Blocks coded
3	32×32	5	17.13	31
2	16×16	5	24.21	275
2	8x8	7	25.84	1410
1	4×4		25.84	3336
0.52 bpp PSNR: 34.34 dB				

3.10.5 Simulation Five

Figure 3-16 and 3-17 are the reconstructed "Lena" corresponding to the search radius of 4 and 2 times of target block size. Because of the small search radius, the quality of reconstructed image is unacceptable. Fig. 3-16 and 3-17 show that we have very poor image subjective quality by using only library blocks in the very close neightbourhood of the target blocks. Although most of the parts are well coded, those blocks that are not well coded result in overall poor quality.

In our new fast coding presented in section 3.8, coding speed is tremendously increased by exploiting self-affine self-similarity with a minimum of library blocks. Only the six affine shapes which are frequently used(see 3.2.1.3).

Fig. 3-18 is the reconstructed image which uses our fast coding scheme by exploiting self-affine self-similarity in the very local neighbourhood. The reconstructed image has good image subjective quality at a low bit rate. "Lena" is coded at 0.45 bpp with a PSNR of 32.4 dB. Reconstructed "Lena" in Fig. 3-19 is coded at the bit rate of 0.35 bpp with a PSNR of 31.4 dB.

Comparing Table 3.12 and 3.15, we notice that the fast scheme can achieve much better coding efficiency at the slight cost of bit rate and PSNR. All these simulations show that our proposed new fast scheme is promising.

Fig.3-16 Search radius 4 times of target block size

Fig.3-17 Search radius 2 times of target block size

Fig.3-18 "Lena" coded by fast coding scheme (Table 3.13)

Fig.3-19 "Lena" coded by fast coding scheme (Table 3.14)

Table 3.13 "Lena" coded by the fast scheme

Image Coded:"Lena" Size: 512×480 8 bpp				
LEVEL	Target size	Threshold	Entropy	Blocks coded
2	32×32	3	14.55	212
1	16×16	9	18.94	2155
0	8x8		20.06	3348
0.45 bpp SNR: 32.4 dB Fig. 3-18				

Table 3.14 "Lena" coded by the fast scheme

Image Coded:"Lena" Size: 512×480 8 bpp				
LEVEL	Target size	Threshold	Entropy	Blocks coded
2	32×32	6	14.55	212
1	16×16	12	18.94	2545
0	8x8		20.06	1788
0.35 bpp SNR: 31.4 dB Fig. 3-19				

Table 3.15 "Lena" coded by the fast scheme

Image Coded:"Lena" Size: 512×480 8 bpp				
LEVEL	Target size	Threshold	Entropy	Blocks coded
3	32×32	5	14.55	17
2	16×16	5	18.94	291
1	8x8	7	20.06	1409
0	8x8		19.61	3992
0.46 bpp PSNR: 32.41 dB				

Chapter 4

Fractal Block Coding With DCT

4.1 Introduction

An image codec using the fractal block coding technique is typically asymmetric. Since a large amount of calculation is involved in searching through library blocks to find the optimal transformation codes in coding, coding is much more computationally costly than decoding. A library block classification was introduced in [9] which can reduce the number of library blocks to search through for each target block so as to increase coding efficiency. However classification itself introduces an extra computational burden. Furthermore, due to the limited classifications, we still have to search through a large number of library blocks to obtain optimal transformations in coding most of the target blocks. The asymmetric problem is partly relieved by the new coding scheme proposed in this chapter.

Multi-level fractal block coding scheme combines different target block size levels to further exploit self-similarity within an image. Smoother areas are coded with larger target blocks while areas with finer details are coded with smaller target blocks. This enables the coder to code an image with different size target blocks adaptively

according to the level of complexity. Therefore images coded by the multi-level block coding technique can be uniformly close to the original image which is crucial for the proposed hybrid codec we introduce in this chapter. In this chapter we first exploit the relation between image resolution and the target block size used in coding. Fractal zooming is done by enlarging the corresponding addresses of target block and library at the same rate. Therefore, we can have as big resolution of decoded image as we like. By the introduction of fractal zooming, we can have a higher resolution image which gives a high effective compression ratio. A small number of fractal codes can retrieve a high resolution image with the loss of high frequency details. We propose a hybrid of fractal block coding and discrete cosine transform for coding higher resoultion images. In comparison with pure fractal block coding technique, the proposed hybrid scheme gives a higher compression with good image quality.

4.2 Fractal Decoding By Zooming

After an image is coded by the fractal block coding scheme, the addresses of the target block and its corresponding libray block are kept, together with other transformation codes. If we want to reconstruct an image, we simply use these codes to iterate on any image to generate a sequence of images which converges to an approximation of the original. We can have fractal decoding by zooming by enlarging

the address of target blocks and their corresponding library block address. The bigger the enlargement, the higher the resolution of the image and the fuzzier the reconstructed large image. A higher resolution image can be generated from the fractal codes for a small image. The higher the zoom rate, the fuzzier the reconstructed image we have.

Figure 4-1 is the original "Miss America" with size of 128x128 at 8 bpp. Figure 4-2 is a reconstructed "Miss America" at the original image size of 128 × 128. Figure 4-3 is a zoomed up version with a zoom factor of two in both x and y directions. Figure 4-4 is a zoomed up version with a zoom factor of four in both x and y directions.

From these figures we can see that the image in figure 4-2, which is decoded at the same resolution of the original image has the best image quality. The one zoomed up by a factor of two has good image fidelity while the one zoomed up by four reveals more spurious image details. Decoding by zooming from the fractal codes of a low resolution image gives you a very high effective compression ratio. However, the higher the zoom factor, the more the spurious details we have.

Fig. 4-1 Original "Miss America" 128x128 with 8 bpp

Fig. 4-2 Reconstructed "Miss America" at original size 128x128

Fig. 4-3 Reconstructed "Miss America" 256x256 with zoom factor 2

Fig. 4-4 Reconstructed "Miss America" 512x512 with zoom factor 4

4.3 Equivalence of Coding Levels

Coding a low resolution image with smaller target block size will significantly increase the coding efficiency. We can obtain a zoomed up version which gives a higher effective compression ratio. You may wonder whether the zoomed up one has the same quality of the images reconstructed by codes using a bigger target block. We will show that coding a higher resolution image at a higher coding level is equivalent to coding a smaller resolution image with lower coding level.

Reconstructed "Miss America" 256x256 coded at target block size 8×8 is shown in Figure 4-5. We also code 128x128 "Miss America" at target block size 4x4. Figure 4-3 is the reconstructed 256x256 "Miss America" with zoom factor 2. Comparing Figure 4-3 and Figure 4-5, we notice that they have very similar image quality and bit rate. However coding a 128×128 image with a target block size of 4×4 only needs about a quarter of the calculation used by coding 256×256 image with a target block size of 8×8. It is much more efficient to code a subsampled version of the original image at a smaller target block size level than coding the original image at a bigger target block size level. Therefore, if we have a very large resolution image, it is more efficient to code all the higher coding levels by coding a very low resolution image.

Fig. 4-5 Reconstructed "Miss America" at the original resolution of 256x256

4.4 Introduction of Discrete Cosine Transform

We have noticed from Figure 4-4 that after zooming by the factor 4, the zoomed up image becomes fuzzy and fine details are lost. However the zoomed up one is still a good approximation to the original image which mainly contains low frequency information. By zooming we achieve a very high effective compression ratio. We attempt to introduce the discrete cosine transform to restore the high frequency detail information lost.

Among the existing transform based methods, the discrete cosine transform(DCT) is near optimum. It is most widely used, both because of the decorrelative property of the DCT and the existence of a fast algorithm for computing the DCT. The DCT can be considered to be a frequency-based operation, transforming 2-D image data into a 2-D cosine frequency domain. Using this new representation for the image, it is easier to discriminate between important and redundant information.

In the two dimension case, the DCT is defined as

$$P(u,v) = \frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} p(x,y)\cos\left[\frac{\pi u(2x+1)}{2N}\right]\cos\left[\frac{\pi v(2y+1)}{2N}\right]$$

$$p(x,y) = \frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)P(u,v)\cos\left[\frac{\pi u(2x+1)}{2N}\right]\cos\left[\frac{\pi v(2y+1)}{2N}\right]$$

with x = 0,1,2, ..., N-1

y = 0,1,2, ..., N-1

where x,y = spatial coordinates in the pixel domain

u,v = coordinates in the transform domain

$$C(u),C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u,v = 0 \\ 1 & \text{otherwise} \end{cases}$$

The transformation kernels are separable. This fact allows the 2D DCT to be conveniently performed in two steps, each of which involves a 1D DCT defined as

$$P(u) = Ap(y)$$

where A is the $N \times N$ fixed transformation matrix

$$[A] = \left[\sqrt{\frac{1}{N}}C(u)\cos\frac{(2n+1)u\pi}{2N}\right], \quad k,n = 0,1,\ldots,N-1.$$

4.4.1 Coding of DCT Coefficients

We are doing a DCT transform on 8×8 block size. The top-left corner of the cosine frequency domain can be considered to be the low frequency region. The bottom-right corner of the block cosine domain are mainly comprised of high frequency coefficients. Redundant information is removed by setting small coefficients to zero which is called thresholding. Non-zero coefficients after thresholding are then uniformly quantized.

4.4.2 Zig-Zag Technique

In order to increase the efficiency of capturing the non zero components a zig-zag scanning class has been adopted:

ZIG - ZAG SCANNING

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

Transmission of the coefficients stops when the last non zero coefficient has been reached.

4.4.3 Coding Of The Scanned Coefficients With a 2D VLC

To increase coding efficiency a two dimensional variable length code has been adopted. This means that "events" are coded. "Event" is defined as:
event : a combination of a magnitude (non-zero quantization index) and a RUN (Number of zeroes defining the end of the run-length are considered as composite rather than separate statistical event)

The run-length and magnitude of composite events define the entries of the 2-D VLC table which contains the code words for composite events.

These codes consists of the following three parts:

1. Escape (6 bits) for indicating the use of fixed length codes.
2. Run (6 bits)
3. Level (8 bits)

After the last non-zero coefficient an End-Of-Block (EOB) marker is sent indicating that all other coefficients are zero. The length of EOB word is two bits.

An example of the two dimensional VLC is shown as follows:

EVENT = (RUN,LEVEL)

Example: (0,3) (1,2) (7,1) EOB

3	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

That means:

* (0,3) The DC component which has the value +3
* (1,2) 2 is next non-zero component according to the zig-zag scanning;the number of zeroes is 1.
* The next component is 1 preceded by 7 zeroes, result (7,1)
* EOB is an End of Block marker which indicates that there are no more zero components.

4.4.4 Huffman Coding

Since events are not uniformly distributed, we encode these events by a technique called *Huffman coding*. The *Huffman coding* gives a variable-length code for each event, where highly probable events are represented by small-length codes, and vice versa. In addition, Huffman codes have a prefix property which means that no short code group is duplicated as the beginning of a longer group. This means that if one character is represented by the bit combination 100, then 10001 cannot be the code for another letter since in scanning the bit stream from left to right the decoding algorithm would interpret the 5 bits as the 100 bit configuration character followed by a 01 bit configuration character.

The *Huffman coding* algorithm is as follows:

1. Arrange the symbol probabilities p_i in a decreasing order and consider them as leaf nodes of a tree.
2. While there is more than one node:
   - Merge the two nodes with smallest probability to form a new node whose probability is the sum of the two merged nodes.
   - Arbitrarily assign 1 and 0 to each pair of branches merging into a node.

Table 4.1 Huffman Code Book of EVENTS

EVENT	Huffman Code	Length(bit)
0,1	110	3
0,2	01000	5
0,3	001010	6
0,4	00001100	8
0,5	001001100	9
0,6	001000010	9
0,7	00000010100	11
0,8	0000000111010	13
0,9	0000000110000	13
0,10	0000000100110	13
0,11	0000000100000	13
0,12	00000000110100	14
0,13	00000000110010	14
0,14	00000000110000	14
0,15	00000000101110	14
1,1	0110	4
1,2	0001100	7
1,3	001001010	9
1,4	00000011000	11
1,5	0000000110110	13
1,6	0000000101100	14
1,7	00000000101010	14
2,1	01010	5
2,2	00001000	8
2,3	00000010110	11
2,4	0000000101000	13
2,5	00000000101000	14
3,1	001110	6
3,2	001001000	9
3,3	0000000111000	13
3,4	00000000100110	14
4,1	001100	6
4,2	00000011110	11
4,3	0000000100100	13
5,1	0001110	7
5,2	00000010010	11
5,3	00000000100100	14
6,1	0001010	7
6,2	0000000111100	13
7,1	0001000	7
7,2	0000000101010	13
8,1	00001110	8
8,2	0000000100010	13
9,1	00001010	8
9,2	00000000100010	14
10,1	001001110	9
10,2	00000000100000	14
11,1	001000110	9
12,1	001000100	9
13,1	001000000	9
14,1	00000011100	11
15,1	00000011010	11
16,1	00000010000	11
17,1	0000000111110	13
18,1	0000000110100	13
19,1	0000000110010	13
20,1	0000000101110	13
21,1	0000000101100	13
22,1	00000000111110	14
23,1	00000000111100	14
24,1	00000000111010	14
25,1	00000000111000	14
26,1	00000000110110	14

The Huffman code can be developed through the utilization of a tree structure as illustrated in Figure 4-6. Here, the symbols are first listed in descending order of frequency of occurrence. The groups with the smallest frequencies(E_3 and E_4) are combined into a node with a joint probability of occurrence of 0.25. Next, that node is merged with the next lowest probability of occurrence symbol or pair of symbols. In this illustration, the pair E_3E_4 is merged with E_2 to produce a node whose joint probability is 0.4375. Finally, the node representing the probability of occurrence of E_2, E_3 and E_4 is merged with E_1, resulting in a node whose probability of occurrence is unity. This master node represents the probability of occurrence of all four characters in the character set. By assigning binary 0's and 1's to every segment emanating from each node, one can derive the Huffman code for each character. The code is obtained by tracing from the 1.0 probability node to each character symbol, noting the 1's and 0's encountered.

Huffman codes can be developed by employing a tree structure. The Huffman code resulting from this construction method is derived by tracing from the 1.0 probability to each source node.The preceding algorithm gives the *Huffman code book* for any given set of probabilities. Coding and decoding is done simply by looking up values in a table.

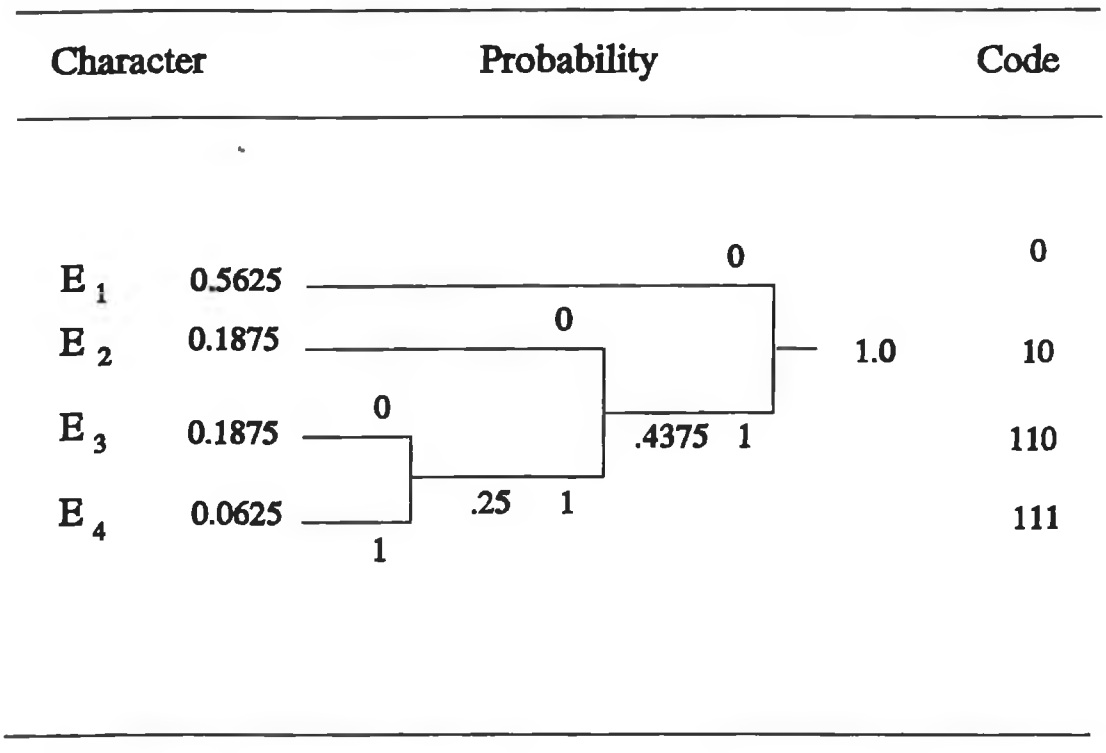


Fig.4-6 Developing a tree structure for four events

In our generation of Huffman codes for EVENTs, only 64 EVENTs which have higher frequency of occurrence are coded. All those events not included in the table are coded by a fixed length of bits.

4.5 Hybrid Codec of DCT and Fractal Block Coding

The flowchart of the proposed hybrid codec is illustrated in Figure 4-7.

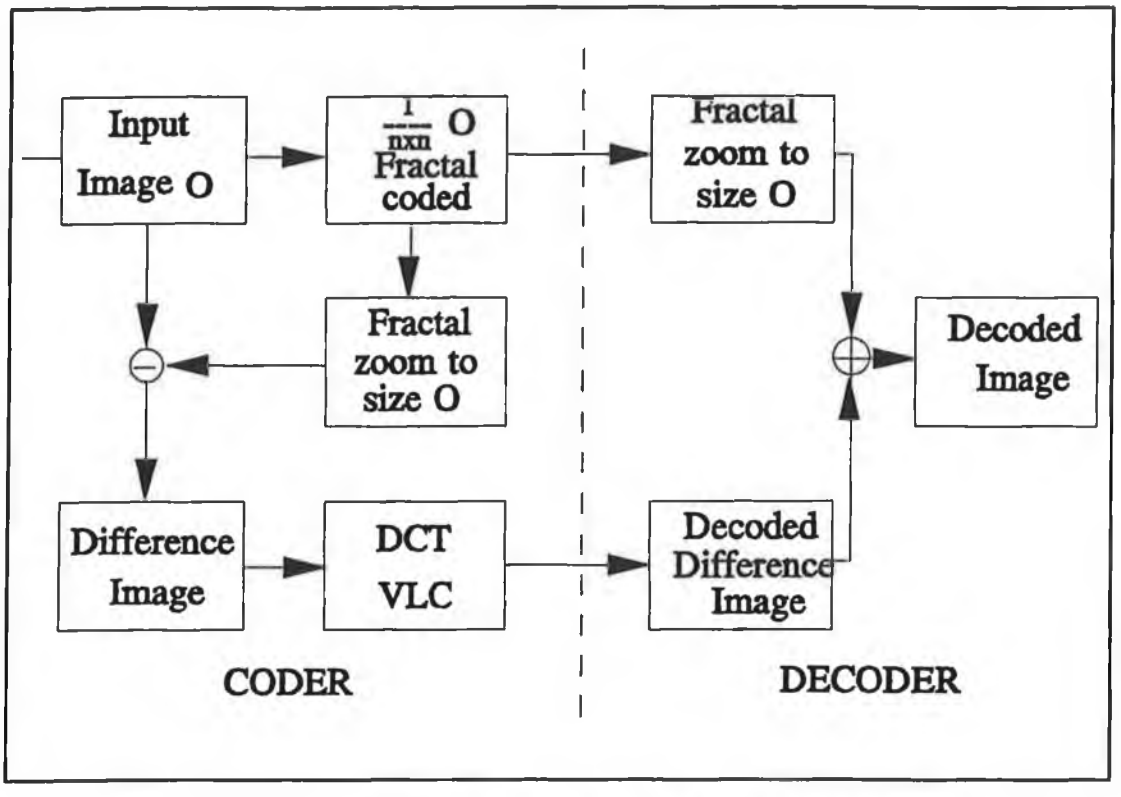


Fig. 4-7. Flowchart of hybrid CODEC

The coder has the following main steps:

1.Input original image O (256×256) is first down sampled to (64×64)

2.1/16 O is then coded by multi-level fractal block coding.

3.Decoded by fractal zooming up to original image size.

4.A difference image between original and the zoomed up one is created.

5.Difference is coded by DCT, the DCT coefficients are variable length coded(thresholding,quantization,zig-zag scanning,Huffman coding).

The decoder simply adds the image generated by by fractal zooming and the decoded difference image to restore the coded image.

Coding a subsampled original by multi-level fractal block coding and decoding by zooming to original resolution gives a very high effective compression ratio. The picture generated by fractal zooming is a good approximation to the original image which will leave a relatively low amount of information to be coded by DCT. Due to the advantage of multi-level fractal block coding, the decoded image is uniformly close to the original image. Therefore difference data left for DCT coding are mainly of very high frequency information which can be well coded by DCT. We can trade subjective image quality against bit rate by the control of DCT thresholds.

4.6 Simulation Results

Simulations are carried out on 256x256 "Miss America" and 512x480 "Lena". Coding results are listed in table 4.2 and 4.3.

Simulation results shows that the proposed hybrid codec is a very efficient codec. By choosing different DCT thresholds, we can trade image quality against bit rate. We have got better simulation results compared with the coding using pure fractal block coding technique.

By using pure DCT coding on 512x480 "Lena" we can achieve 0.28 bpp at a PSNR of 31.33 dB. Our hybrid scheme is slightly better than pure DCT. Our hybrid scheme can be further improved with the advancement of fractal coding techniques.

Table 4.2 Simulation results of "Miss America"

"Miss America" 256x256 with 8 bits per pixel				
figure	fractal ratio	DCT threshold	bit rate(bpp)	SNR(dB)
4-8	103.2 : 1	16	0.36	37.0
4-9	103.2 : 1	24	0.23	35.6
4-10	103.2 : 1	32	0.19	34.7

Table 4.3 Simulation results of "Lena"

"Lena" 512x480 with 8 bits per pixel				
figure	fractal ratio	DCT threshold	bit rate(bpp)	SNR(dB)
4-11	81.0 : 1	24	0.37	33.4
4-12	81.0 : 1	40	0.25	31.6

4.7 Alternate Combination of Fractal Block Coding and DCT

In this chapter we have presented a hybrid algorithm by using DCT coding on the difference image after fractal block coding. Then readers may wonder how about first DCT than fractal coding. In theory, fractal block coding scheme is successfull while coding image with natural scenes where block-wise self-similarity do exist. Difference image after DCT coding is no longer an image of natural scene.

We first code 352x288 "Miss America" with DCT coding with 0.43 bpp at a PSNR of 41.14 dB (threshold 16). Than we use fractal block coding on the difference image we get 1.18 bpp. We have a PSNR of 14.64 dB with the decoded image which is a severely distorted and totally unrecognizable. This is simply because of the limitation of fractal block coding scheme. Artificial images without block-wise self-similarity are not suitable for fractal block coding scheme.

Fig. 4-8 "Miss America" 256x256 coded with DCT threshold 16

Fig. 4-9 "Miss America" 256x256 coded with DCT threshold 24

Fig. 4-10 "Miss America" 256x256 coded with DCT threshold 32

Fig. 4-11 "Lena" 512x480 coded with DCT threshold 24

Fig. 4-12 "Lena" 512x480 coded with DCT threshold 40

Chapter 5

Applications of Fractal Block Coding to Video Compression

Video and image compression are essential technologies in a video-oriented world where new applications are being proposed with significant volume production opportunities. The need for video compression occurs in a wide range of applications: for transmission cost reduction for video phones and video conferencing systems, to reduce the bandwidth and power required to transmit data from remote sources such as satellites, and to provide compact interactive video for training and entertainment from a standard-size CD. The primary motivation behind image compression is the need to reduce the amount of information that must be stored on tape or disk to achieve an image. In a desktop publishing environment, compression can increase the number of high-quality images that can be stored on a floppy disk from approximately 2 to 40, without noticeable degradation.

To be fully aware of the application value our fractal video codec, we first introduce the impending video telephony as an ISDN application. Our video codec using pure fractal block coding is first presented and compared with that of H.261. Secondly there is the hybrid fractal block coding and DCT video codec. Simulations of both video

codecs are presented.

5.1 Contemporary Image Compression Standards

Several compression algorithms are in the process of becoming international standards, each of which is suited to different applications. The proposed Joint Photographic Expert's Group (JPEG) standard [28] specifies a method of compressing still images. The primary mechanisms of compression are the frequency-dependent quantization of DCT coefficients and subsequent Huffman coding of nonzero quantized coefficients. In JPEG, there is no notion of a time variable. However, JPEG can be used in a video environment by ignoring the temporal dependencies. The CCITT H.261 standard [72] proposed for video telephony, is similar to JPEG in that DCT coefficients are quantized and coded with a Huffman coding algorithm. The primary difference between H.261 and JPEG is the use of motion compensated temporal prediction in H.261. Essentially, instead of coding the image directly, the temporal prediction errors are coded. For video storage, the Motion Picture Expert's Group (MPEG) standard [75] has been proposed. MPEG and H.261 are similar, but MPEG provides greater flexibility and the ability to achieve greater compression with the same quality. These features come at the expense of increased complexity.

5.2 Video Telephony As An ISDN Application[60]

Visual telephony is the process of communicating at a distance by using moving images as well as voice. Today a telephone conversation is virtually second nature to all people in developed nations, but the usage of video telephony is still a very rare occurrence. The reason for this situation are many. Several key factors are elaborated below.

A voice signal can be transported through a telephone network with a bandwidth of 3.4 kHz while a video signal, using a regular television signal as an example, requires a bandwidth of 4.3 MHz, a ratio of 1:1265. Even though a picture is worth a thousand words, almost no one would want to pay for a visual conversation costing a thousand times more than a telephone call, assuming that connections are tariffed on the basis of bandwidth usage. Consequently, bandwidth reduction is the necessary first step to bringing visual telephony closer toward reality.

There are many different ways to reduce the bandwidth required to transport a video signal. A great deal of progress has been made in the last two decades. Some techniques employ analog methods, but most employ digital coding techniques. It is important to note that the coding algorithm implemented for reduced bandwidth transmission has to be known to the receiving party, so that the underlying video signal can be decoded to its original form. Although various kinds of video

codecs(coder/decoders) exist today, they cannot communicate with each other because they employ different coding algorithms. Consequently, the second important step for a successful realization of video telephony is that the compression technique has be to standardized.

Due to the tremendous complexity of compression algorithms, a great deal of sophisticated electronic circuitry is required for their implementation. This is the major reason why video codecs are very expensive today. For ubiquitous deployment of any future video service, the cost of codecs has to be drastically reduced. Therefore, the next important step is to attain significant cost reduction by employing VLSI technology.

The last, but not the least, important step is the availability of a transport network to provide such a service. This network has to be digital, of high capacity, full duplex, switchable, and ubiquitous. The timing of the ISDN(Integrated Service of Digital Network) deployment appears to be quite appropriate for the application of visual telephony.

Every step mentioned above, compression, standardization, VLSI and network availability, is essential for the successful deployment of new low bit-rate video services. Past attempts in providing similar services, such as Picturephone and Picturephone Meeting Services, were not successful because of the fact that one or

more of the above mentioned ingredients were missing. The situation today is drastically different from the past. It appears that video telephony using ISDN will soon become a reality.

5.2.1 Video Telephony System Overview

A system block diagram for video telephony is shown in Figure 5-1. With the end-to-end digital connectivity of ISDN, a user is able to call another user with any combination of video, audio, and data at various rates. The transmission coder will combine video, audio and data, together with framing and other information, and deliver the resultant bit stream to an ISDN access port. The transmission decoder will perform the inverse operation. At present, there are two kinds of ISDN access that have been defined: Basic Access, and Primary Access. Basic Access supports 2B + D channels where the bit-rate for a B channel is 64 kb/s and that of a D channel is 16 kb/s. Basic Access will be available to every household and business when ISDN is fully deployed. Due to the severely limited available bit-rate, Basic Access is suitable only for desk top fact-to-face visual communication which is often referred to as videophone. Primary Access supports 23B+D channels where the bit-rate for both B and D is 64 kb/s. Since primary Access requires a T1 line it is expected to be used mostly in business applications. Due to the additional available bit-rate, pictures transmitted by Primary Access can be more complex with better quality. Primary

Access is therefore more suitable than Basic Access for video teleconferencing. In summary, for ISDN applications, the bit-rate for combined video and audio services is limited to $p \times 64$ kb/s where $p = 1,2,..., 30$(for North American network, $p = 1,2,...,23$). The CCITT Specialists Group on Coding for Video Telephony has been working on the standardization of a video coding algorithm for these rates. An international standard, H.261, was established in mid 1990.

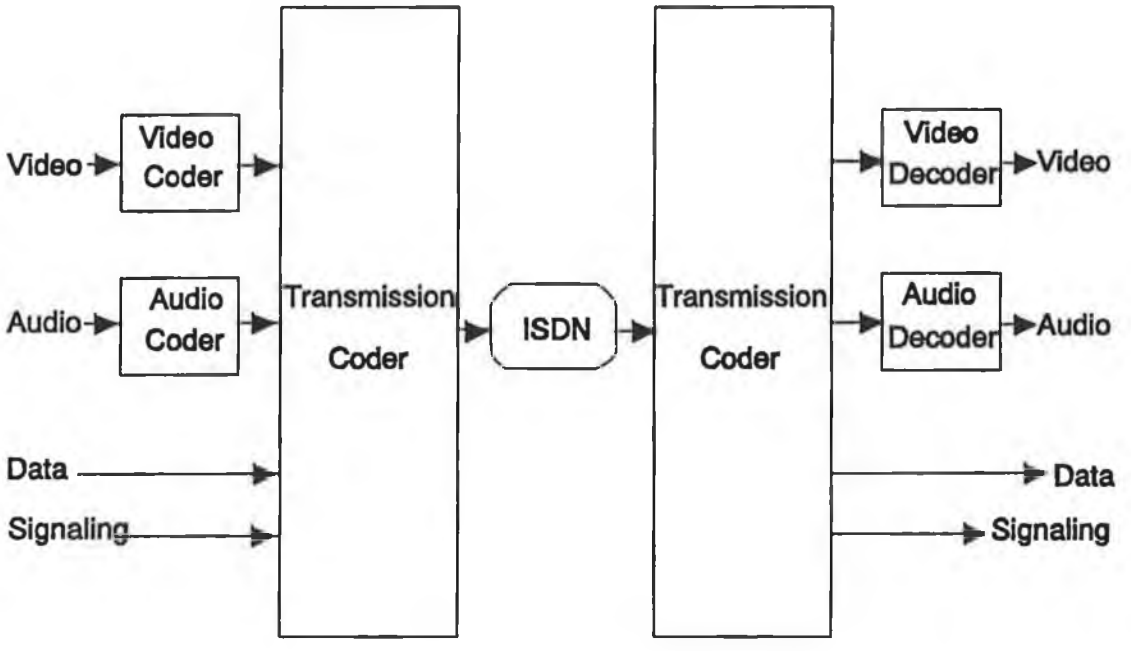


Fig. 5-1 Video Telephony System Architecture

5.2.2 Introduction to CCITT H.261 Standard

The CCITT H.261 Standard specifies a method of communication for visual telephony. It is sometimes called the $p \times 64$ standard, because the channel data rate can be

chosen to be p × 64 KBits/Sec. If p = 1, then a low quality video signal for use in picture phones can be transmitted over a 64 KBit/Sec ISDN line. If p = 32, a high quality video signal for teleconferencing can be transmitted over a 2 MBit/Sec line.

The standard itself specifies exactly the organization and interpretation of transmitted bits to make it possible for two codecs from different manufactures to communicate successfully. In addition, the computations for part of the algorithm have also been specified to ensure that the predictors used in the encoder and decoder track. In general, the reconstruction loops(essentially the entire decoder) have been specified in a fairly detailed manner, leaving little flexibility. On the other hand, all operations outside this loop are not specified. With this freedom, the system designer can choose various cost/performance tradeoffs by adding novel features or by simply not performing certain operations. To illustrate these points the IDCT operation is specified in a very detailed way, while the DCT is not specified at all. A compatible system does not actually need to use a DCT, but it is recommended.

The format of internal representation of the image has been specified to be either CIF(optional) or QCIF(mandatory). CIF luminance images are 352 pixels × 288 lines, while QCIF luminance images are 176 × 144. The chrominance images are half resolution vertically and horizontally. The nominal frame rate is approximately 30 Hz, but the encoder can choose not to encode all frames to effectively reduce the frame rate. This format is used only internally and does not imply any particular format for

the final displayed video. CIF was chosen because it can be easily converted to and from both NTSC and PAL formats to make use of readily available cameras and monitors.

The CCITT H.261 encoder is a hybrid coder based on transform coding and temporal predictive coding with motion compensation. Most operations are performed on macroblocks or blocks. Each macroblock consists of four 8×8 luminance blocks and two 8×8 chrominance blocks. Each chrominance block occupies the same physical area as the four luminance blocks. The chrominance blocks have reduced resolution because the human visual system is less sensitive to colour detail than luminance detail. The encoder has two modes of operation. The intraframe mode realizes compression within one frame, in the spatial dimension only. Compression in the temporal direction is disabled. This mode is used for coding the first frame in a scene and for resetting the prediction loop. The interframe mode, in addition to compression within one frame, also realizes compression between consecutive frames. The previous decoded frame is motion compensated and then used as a prediction for the current frame.

5.3 Video Codec One: Multi-level Fractal Block Coding

Simulations of multi-level fractal block coding in Chapter 3 and 4 demonstrated that

fractal block coding is a promising coding scheme for individual images. We now want to introduce it to a video compression application. A video codec using multi-level fractal block coding is presented and simulation results are given. The proposed video codec is basically an intra-codec which means interframe correlations are not exploited. This will be discussed and explained later.

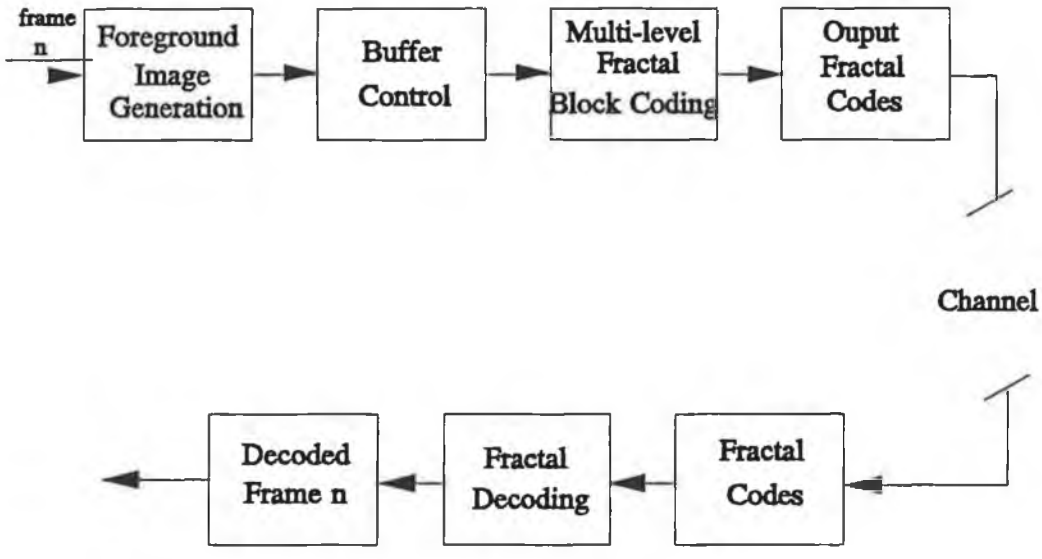


Fig. 5-2 Video Codec One: Pure Fractal Block Coding Techniques

When applied to video coding, we only need to code parts with significant movement instead of coding the whole of each image in a sequence. In typical head and shoulder video sequences, only the head and shoulder parts are moving from frame to frame. The background often remains virtually unchanged. We refer to the parts of the

picture with detected movement as *foreground image*, and this is the part which will be coded by multi-level fractal block coding. The background image is left uncoded.

The proposed video codec illustrated in Figure 5-2 is a direct application of multi-level fractal block coding scheme. Its coder is basically an intra-frame codec which is mainly comprised of the following:

- Foreground Image Creation
- Buffer Control
- Multi-level Fractal Block Coding of Foreground Image
- Quantization

5.3.1 Generation of Foreground Image

Foreground image is comprised of 16×16 size blocks of the new frame which are significantly different from that of the previous frame. If the mean square error of a 16x16 block of the difference between the corresponding new frame and the previous frame is bigger than an error threshold, the 16×16 block of the new sequence is considered as significantly different and it becomes part of the *foreground image*. By choosing a 16x16 block size in foreground generation, we can exploit the affine-block-wise self-similarity of head and shoulder video sequences at target block sizes of 8×8 and 4×4.

5.3.2 Buffer Control

According to the size of the *foreground image* and the availability of data bandwidth, a buffer control mechanism is used to decide the target block size levels and allocate the number of target blocks for corresponding levels. In case of a shortage of data bandwidth, more blocks are coded at larger target block sizes while more smaller target blocks are used when we have sufficient bit rate available. Under this buffer control scheme, the bits available are used to the fullest extent. Because of the buffer control mechanism introduced, the setting of a threshold for higher level target block size is totally omitted by simply choosing the amount of best coded target blocks at that level decided by buffer control.

The detailed buffer control algorithm is as follows:

Assuming we have following information

BitsAvail	(Calculated from the available transmission rate)
ForeSize	(Number of 4×4 block in foreground image)
BPB_4	(Bits per block for target block size 4×4)
BPB_8	(Bits per block for target block size 8×8)
N_4	(Number of target blocks coded at 4×4 level)
N_8	(Number of target blocks coded at 8×8 level)

$$\begin{pmatrix} BPB_8 & BPB_4 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} N_8 \\ N_4 \end{pmatrix} - \begin{pmatrix} BitsAvail \\ ForeSize \end{pmatrix}$$

N_8 and N_4 change according the transmission bitrate and the size of the foreground image. The strategy is to code the maximum number of blocks at a lower target level so as to produce the best quality.

5.3.3 Decoder

Decoding is exactly the same as still image decoding. Decoding convergence normally needs only about 2 iterations. That's because we have a pretty good starting image which is close to the next sequence. The proposed video codec remains asymmetric with coding more computationally costly than decoding.

5.4 Comparison with H.261

Our video codec is basically an intra-codec which means inter-frame correlation is not exploited. In our scheme, image self-similarity is searched within the foreground image. The reason for this is that image affine-block-wise self-similarity exists in the very local area of a target block in the foreground image which still contains mainly head and shoulder parts. Therefore it is unnecessary to use library blocks from previous frames which requires more memory. The advantage of this is that the

foreground image could be generated without referring to other frames. The disadvantage of this is that the bit rate requirement remains high.

In the CCITT H.261 video codec, motion prediction plays an important part in removing inter-frame redundancies. DCT coding the difference image after motion prediction allows us to achieve a low bit rate with good image quality.

However motion prediction can't be introduced into our video codec. The reason for this is that a fractal codec is only suitable for coding images of natural scenes. The difference image after motion prediction is no longer a natural image, so does not exhibit self-similarity.

It would be nice if we could just keep the motion of corresponding target blocks and library blocks instead of sending a whole group of codes for a target block. It seems possible at first glance. However, if you analyse in detail, you will find that it is not possible. First, we recall fractal decoding starting from a uniform greytone image. Each target block uses the corresponding library block to transform and cover itself. However this library block possibly contains four smaller target blocks which are covered by different library blocks from different parts of the image. Therefore a target block has a connection with possibly many library blocks. To trace the motion of a target block itself is not difficult. However, tracing the motion of a library block results in tracing motion vectors of many other related library blocks. In conclusion,

removing redundancies by tracing the motion vector of a target block and its library block is not practical.

Coding a target block on an incoming new sequence by using library blocks from previous frames is not a viable operation for fractal coding. A fractal block coding scheme is supposed to exploit self-similarity. Decoding is done without using any information out of the coded image. Removing interframe redundancies can be better done by motion compensation techniques[41,42].

5.5 Simulations of Video Codec One

Our simulations are carried out on the QCIF(176×144) greyscale "Miss America" and "Salesman" images which are typical head and shoulder test sequences. Since these head and shoulder images are so condensed, the smallest target block size is set at 4×4 which will give good image quality in the high frequency areas. The largest target block size is chosen as 8×8 which is used for the smoother areas of face and shoulder. Hence, their corresponding library sizes have to be chosen as 8×8 and 16×16.

Figure 5-3 shows four frames of the original quarter CIF sequence of "Miss America". Reconstructed frames of "Miss America" are shown in Figure 5-4.

A frame rate of 10 frame/s is chosen for all the simulations of the two video codecs. When coding "Miss America" using video codec one, a transmission rate of 96 kbit/s is achieved with average SNR of 36.3 dB.

Figure 5-5 shows four frames of the original quarter CIF sequence of "Salesman". Reconstructed frames of "Salesman" are shown in Figure 5-6. A transmission rate of 192 kbit/s is achieved with average SNR of 30.6 dB. We notice that simulation results for "Miss America" are much better than that of "Salesman" even at half the bit rate. The reason for this is that "Salesman" images are much more complicated than "Miss America" images.

Fig. 5-3 Four Frames of The Original QCIF "Miss America" Video Sequence at 10 frames/s

Fig. 5-4 "Miss America" Video Sequence Coded

By Video Codec One at 96 Kbit/s

Fig. 5-5 Four Frames of The Original QCIF "Salesman" Video Sequence at 10 frames/s

Fig. 5-6 "Salesman" Video Sequence

By Video Codec One at 192 kbit/s

5.6 Video Codec Two: Fractal block Coding and DCT

The proposed video codec with combination of DCT with fractal block coding techniques is shown in Figure 5-7.

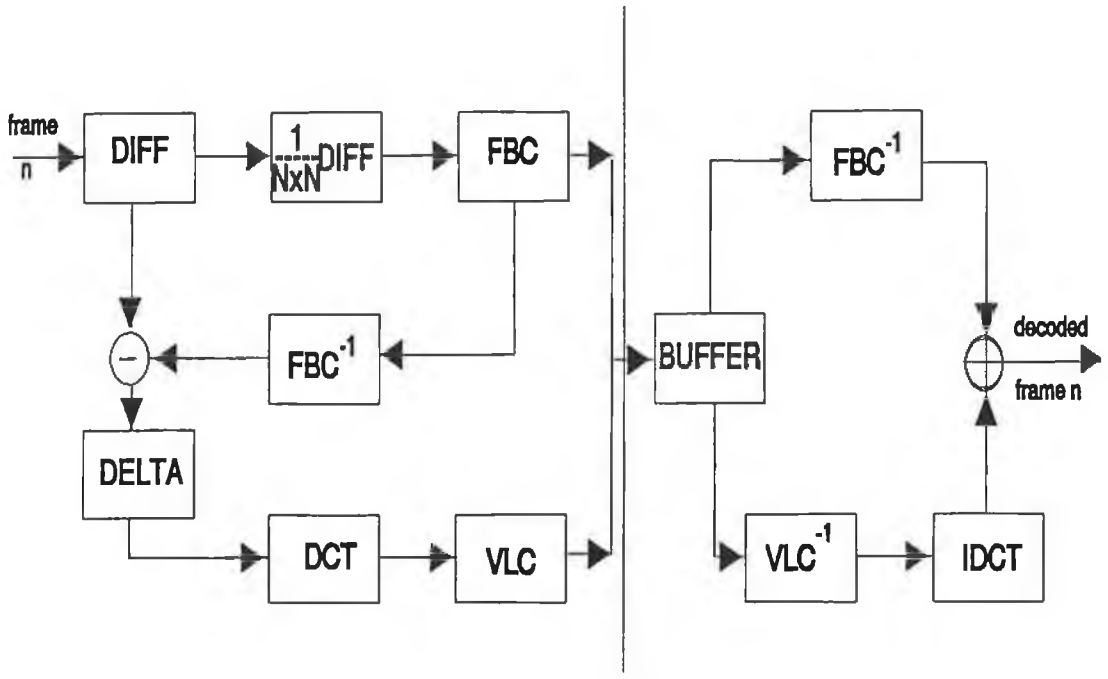


Fig. 5-7 Video Codec Two: Hybrid DCT/Fractal Block Coding

DIFF = foreground image

1/(N×N)DIFF = a subsampled version of DIFF

FBC = fractal block coding 1/4DIFF

FBC^{-1} = fractal decoding with zooming

DELTA = difference image of DIFF and FBC^{-1}

DCT = discrete cosine transform

IDCT = inverse discrete cosine transform

VLC = variable length coding

VLC^{-1} = variable length decoding

5.7 Simulations of Video Codec Two

These simulations are also carried out on QCIF(176×144) greyscale "Miss America" and "Salesman" head and shoulder test sequences.

The reconstructed sequence of "Miss America" coded at bit rate 96 kbit/s is shown in Figure 5-8 with average SNR of 37.53 dB.

Figure 5-9 shows the reconstructed sequence of "Miss America" coded at bit rate of 64 kbit/s. The average SNR is 35.8 dB.

Figure 5-10 shows the reconstructed "Sales" sequence at 192 kbit/s with a SNR of 32.38 dB. The reconstructed "Sales" sequence coded at 128 kbit/s with a SNR of 30.76 dB, is shown in Figure 5-11.

These simulations show that video codec two can achieve a lower bit rate than video codec one. Video codec two is closer to symmetric which is more suited for real time video telephony applications. Video codec one is better suited for offline applications.

Fig. 5-8 "Miss America" Video Sequence Coded

By Video Codec Two at 96 kbit/s

Fig. 5-9 "Miss America" Video Sequence Coded

By Video Codec Two at 64 kbit/s

Fig. 5-10 "Salesman" Video Sequence Coded

By Video Codec Two at 192 kbit/s

Fig. 5-11 "Salesman" Video Sequence Coded

By Video Codec Two at 128 kbit/s

Chapter 6

Conclusions and Discussion

6.1 Conclusions

Barnsley and Sloan[4] first proposed the idea of image compression by fractal techniques. Their coding scheme is based on the collage theorem with iterated function systems. For an arbitrary black-and-white image, we try to find a group of affine transformations on the original image and cover on the original image reasonably well. As long as the cover is good enough, the collage theorem with iterated function systems guarantees that the iterated sequence provides a good approximation to the original image. For an arbitrary black-and-white image, this turns out to be a very complicated optimisation problem which can't be managed even by the most powerful computer available at present.

To realise a practical application to a colour image, the image is first broken into segments by traditional image processing techniques such as colour separation, edge detection, spectrum analysis and texture-variation techniques. For each segment, we look it up in a library of fractals. This fractal library contains IFS codes for each fractal image instead of holding the actual fractal image. The library is organized in

such a way that IFS codes which generate similar fractal images are grouped together. When each segment finds its corresponding IFS codes in the library, the IFS codes are kept instead of the original image information. These IFS codes can be used to iterate to generate the approximation of the original image.

Barnsley's fractal coding scheme is a very complicated one which requires complex image processing techniques and needs a library of fractal codes for different kinds of segments. The system sounds feasible but practical realisation is a non-trivial task.

Jacquin's fractal block coding scheme is a practical new coding technique which can compresses image data in an automated manner. Image redundancies are removed by exploiting block-wise self-similarity. Unlike Barnsley's scheme, image processing and a library of fractals are not necessary.

The foundation theory for fractal block coding is the collage theorem on image space which is the extension of the collage theorem with IFS. Arbitrary digital greytone images are modelled as fixed points of contractive transformations on the image space. The transformation coefficients are used to iterate successively from any initial image to a fixed point which is close to the original image.

In Jacquin's fractal block coding, the image is first partitioned into two sizes of non-overlapping square blocks. Smaller square blocks are called target blocks while

the larger ones are called library blocks. For each target block, we search through these library blocks to find a library block which can be transformed to cover the target block well. The address of the library block which gives the best cover is kept together with other transformation codes.

During coding each target block, library blocks are taken from the original image. After all the target blocks are coded, the information of these library blocks is no longer needed at the receiver end. We simply keep all these fractal codes and transmit these codes through the channel. At the receiver end, these fractal codes are used to iterate on any initial image which generates a sequence of images converging to an approximation of the original image.

In this dissertation, a multi-level fractal block coding scheme is proposed. This idea lies in the fact that in natural images block-wise self-similarity does not necessarily happen at one scale. It is more efficient to use a different target block size to code image parts with different complexity.

In multi-level fractal block coding, smoother areas are coded with a larger target block size while finer details areas are coded with smaller target blocks. The scheme is able to adjust the size of target block adaptively according to the scale of self-similarity occurring among the image to be coded.

By exploiting image self-similarity at different target block size levels, self-similarity in natural images is exploited to a further extent. Therefore a lower bit rate can be achieved by this scheme. At each higher level, a threshold is set. Those target blocks which can't be coded under the specified thresholds are to be coded at lower coding level. With the control of error thresholds for different coding levels, we can make a tradeoff between bit rate and image quality.

In order to reach low bit rate coding, we hope more blocks can be coded at a higher target block size level. In our simulations, we find that image parts with fine details and especially sharp edges, are difficult to code at a higher target block size level which results in annoying "staircase" effects. This is due to the failure to find a good library block that gives a satisfactory match after exhaustive searching through all the library blocks.

In previous implementations of fractal block coding, redundancies are removed by exploiting block-wise self-similarity. Self-similarity is searched for between a target block and library blocks which are both of square shape. In this dissertation, we introduce the removal of affine-block-wise self-similarity which includes square-block-wise self-similarity as a special case.

In the new scheme, the library blocks are not necessarily square blocks. They can be affinely distorted square blocks in both x and y directions. By taking into account

different orientations and affine transformations of square library blocks, the library is substantially enriched thus increasing the chance of finding a library block for a good match at a higher coding level.

In a previous implementation of fractal block coding[20], the "Lena" image of 512x512 pixels with 8 bits per pixel was coded at 0.5 bpp with a SNR of 30.8 dB. The simulation result of our proposed multi-level fractal block coding with affine-block-wise self-similarity gives 0.52 bpp at 34.34 dB.

Fractal block coding has the reputation of computational complexity in encoding thus inhibiting it from real-time application. Since fractal decoding is very fast, it is possible to apply fractal techniques to offline coding applications like image information retrieval. The other problem with encoding is that the coding time required is uncertain. The only decision to abort searching is immediately after a satisfactory library block is found. If computational complexity can be reduced significantly, the fractal block coding technique will have wider applications in image compression areas.

One practical way of reducing coding complexity is to classify library blocks into certain number of categories like edge blocks, midrange blocks and plane blocks. For each target block, we first identify the category it belongs to and then search the corresponding group of library blocks. Thus time can be saved by avoiding wasting

time trying disimilar library blocks. With the limitation of classification techniques, we still have a large number of library blocks to try for coding each target block. Although part of the search effort is relieved, coding still remains computationally costly.

In this dissertation, we propose a very fast coding scheme which can bring fractal block coding technique closer to real-time applications. In previous implementations of the fractal block coding, searching through a large number of library blocks has always been required to obtain a library block that gives a good match. Such a library block is found somewhere in the original image. Simulations have shown that using target blocks in the very near region only will result in unacceptable subjective image quality. But a very efficient coding scheme is realised in this dissertation by exploiting affine-block-wise self-similarity between a target block and the library blocks which have the target blocks at one of its corners. With the application of affine-block-wise self-similarity, correlations between a target block and its local area library blocks are well exploited. The promising simulations have shown that affine-block-wise self-similarity do exist between a target block and its very close neighbourhood.

This new strategy towards library blocks has brought about two important advantages. Firstly, since similarity is searched in the very local area which use four square library blocks and its associated affine-square-block pieces, coding complexity is tremendously reduced. Secondly, since the library block we use for each target block is in the very

immediate neighbourhood, many bits used previously for describing the library address are saved. Simulation shows that "Lena" (512×480 at 8 bpp) can be coded at 0.35 bpp with a SNR of 31.4 dB.

We have also studied the relationship between image resolution and choice of target block size. We have discovered the equivalence of image resolution and target block size. Simulations have demonstrated that coding an image at a higher resolution with a big target block size is approximately equivalent to coding a small version of image with smaller target block size. We use fractal block coding to give a high quality low resolution image. We then use fractal zooming to give a higher resolution image. The high frequency information in the higher resolution image can be compensated by discrete cosine transform coding. "Lena" (512x480 at 8 bpp) is coded at 0.20 bpp with a SNR of 30.5 dB.

The direct application of our still image codec to video compression shows that the results for a video codec using fractal block coding technique are also promising. With the further improvement of coding efficiency in the future, it can be applied to real time digital video compression. Our simulation on QCIF "Miss America" at frame rate 10 frame/s shows that a bit rate of 128 kbit/s with good image quality can be achieved.

6.2 Discussions for Future Work

The fractal block coding technique is relatively new compared with other image compression techniques. JPEG for still image coding is pushing the limit of its capabilities while fractal image compression techniques can go further. The simulation results presented in this dissertation have demonstrated that fractal block coding technique is a very promising compression scheme. Further reduction in bit rate and improvements in coding/decoding efficiency can be achieved. We will discuss some possible ways of improving our present scheme. The fractal block coding technique can be applied to object-oriented analysis-synthesis coding for very low bit rate video compression.

6.2.1 Triangular Partition

In fractal block coding, image target blocks are of square shape. In our multi-level scheme, the image is encoded at different target square block sizes. If a bigger block fails to be coded at the higher target block level it will be split and attempted to be coded at a lower target block level. The final coded image is a combination of different sizes of square blocks.

We think that it should be more efficient to combine different shapes to give image partitions to generate target blocks. Instead of square image partitions, we can employ a triangular partition. We can start off with square target blocks and then use triangular splitting. At the same splitting level, the size of a triangular piece is bigger than the square shape piece. If a bigger piece can be coded at a higher level, a lower bit rate can be achieved. With block-based coding, a blocking effect is inevitable at low bit rates. With triangular partition, we can improve subjective image quality by taking advantage of the fact that human visual system is less sensitive to distortion in the diagonal direction.

6.2.2 Splitting Strategies

In the multi-level scheme presented in this dissertation, a target block at a higher level has to be split into four lower level target blocks if it fails to be coded within the threshold of that corresponding level. It is quite possible that serious distortion on one or two of the smaller target blocks results in the failure of the whole target block to be coded within the threshold. Part of the larger blocks which are well coded do not have to be coded at a lower level.

This may be better done by checking the coding results of the combinations of the four smaller square pieces. Those combinations which are well coded form a new

shape of the target piece which is a combination of square shapes instead of a single square block piece. Only the parts not well coded are to be coded at a lower coding level. The shape of the irregular target blocks are considered as coded within the threshold of corresponding coding levels. The shape of the irregular piece coded is kept together with the other transformations.

The combination of these target blocks has a bigger image size than each small square target block size. The bigger the irregular target block coded at a higher level, the lower the bit rate we achieve.

6.2.3 Coding Efficiency

Coding efficiency is of great concern in our future research as it is crucial for real time applications. There are some possible ways which can be examined in the future to improve image coding efficiency.

One improvement we can make on our present scheme is to have some initial image analysis on the target block to be coded. The finer the detail in the target block size, the more difficult it is to code it at a higher level. We can make an initial decision of splitting depending on the complexity of the target block to be coded. If a target block is quite complicated, one can try to code it at a lower level. A target piece can be

simply coded by the its average greylevel when it is detected as a smooth piece. By these strategies, time is saved by avoiding trying out in vain the library blocks together with all those combinations of isometric transformations.

6.2.4 Decoding Efficiency

In our scheme of decoding, we iterate on an arbitrary image and generate a sequence of images converging to an approximate image of the original one. Decoding usually takes about 5 iterations to get a reconstructed image. A fast non-iterative decoding algorithm is proposed in [21]. It is shown in the scheme that decoding efficiency can be significantly improved by introducing a modification providing an iteration-free decoder. The coding results are not affected by this modification. The new decoding algorithm requires only 1 multiplication and 3 additions per pixel. Further investigations should be carried out in the future to see whether this fast decoding scheme can be combined with our proposed fast coding scheme.

6.2.5 Application in Future Video Codec

In our current scheme of video coding, frame to frame temporal redundancies are not exploited. To have a more efficient video coding, we need to exploit temporal redundancies fully. We are considering the combination of our fractal block coding

with objected-oriented analysis-synthesis coding[44,45] based on the source model of moving 2-D and 3-D objects which is a promising video coding method for very low bit rate video telephony applications.

In the H.261 video coding scheme, each image of a sequence is usually subdivided into blocks of $N \times N$ picture elements and the luminance and chrominance signals of each block are encoded by motion compensated predictive and transform coding algorithms. Thus an image is described by independently moving square blocks which can lead to visible distortions known as blocking and mosquito effects of low bit rate codecs. To avoid these image distortions more appropriate source models for describing the image have to be introduced. Object-oriented analysis-synthesis coding subdivides each image into objects of different motion. Each object is described by three sets of parameters, namely shape, motion and colour parameters. The latter consists of the luminance and the chrominance information of an object. Objects which comply with the underlying source model are synthesized from previously transmitted colour parameters. The objects which fail the model colour parameters are called model failure areas which are to be coded by intra-frame coding techniques.

We are very interested in applying our fast fractal block coding to model failure objects. Efficient coding of model failure areas is of great importance to very low bit rate coding. At present, model failure areas are coded by vector quantization

techniques. Since, in head and shoulder video telephony applications, these model failure objects are mainly parts of hair, mouth and eyes which are parts of natural images, these model failure objects can be well coded by the fractal block coding technique.

References

1. Mandelbrot B.B. *The Fractal Geometry of Nature.* Freeman, New York 1983.

2. Barnsley. M.F. *Fractals Everywhere.* Academic Press, Boston 1988.

3. Hutchinson, J., "Fractals and self similarity", *Indiana Univ. J. Math.***30**, pp. 713-747. 1981.

4. Barnsley M. F. and Sloan A., "A better way to compress images", BYTE Magazine, January 1988.

5. Barnsley M. F., "Iterated function systems" from "Chaos and fractals the mathematics behind the computer graphics", Proceedings of Symposia in Applied Mathematics, Vol. 39 pp. 127-144.

6. Barnsley M.F. and Demko S., "Iterated function systems and the global construction of fractals", *Proc. R. Soc. Lond.* A **399**, pp. 243-275, 1985.

7. Barnsley M.F. Ervin V. Hardin D. and Lancaster J., "Solution of an inverse problem for fractals and other sets", *Proc. Natl. Acad. Sci. USA,* Vol. 83, 1975-1977, April 1986.

8. Barnsley M.F., "Application of recurrent interated function systems to images", *SPIE Vol. 1001 Visual Communications and Image Processing,* 1988.

9. Jacquin A.E., "Fractal image coding based on the theory of iterated contractive image transformations", *SPIE Vol. 1360, Visual Communications and Image Processing,* 1990.

10. Jacquin A. E., "A novel fractal block-coding technique for digital images", *Proc. IEEE Int. Conf. On Acoustics, Speeach and Signal Processing,* pp. 2225-2228, 1990.

11. Jacquin A. E., "Image coding based on a fractal theory of iterated contractive image transforamtions", *IEEE Trans. Image Processing,* Vol. 1, No. 1, pp. 18-30, 1992.

12. Jacquin A. E., "A fractal theory of iterated markov operators in spaces of measures, with applications to digital image coding", PhD thesis, Georgia Inst. of Technology, 1989.

13.Jacquin A.E., "Image coding based on a fractal theory of iterated contractive image transformations", *IEEE Trans. Image Processing*. Vol. 1. No. 1. January 1992.

14.Beaumont J. M., "Advances in block based fractal coding of still pictures", IEE Colloquium on the Application of Fractal Techniques in Image Processing, IEE Digest No. 1990/171, 1990.

15.Beaumont J. M., "Image data compression using fractal techniques", BT Technol. J., Vol. 9, No. 4, pp.93-109, 1991.

16.Monro D.M. and Dudbridge F., "Fractal approximation of image blocks", *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. III: 485-488, 1992.

17.Monro D. M., "A Hybrid Fractal Transform", *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. V: 169-172, April 1993.

18.Monro D. M., Dudbridge F. and Wilson A., "Deterministic rendering of self-affine fractals", IEE Colloquium on Fractal Techniques in Image Processing, IEE Digest No. 1990/171, 1990.

19.Monro D. M., "Least squares fractal interpolation", Colloquium on Fractals in Engineering, Ecole Polytechnique, Montreal, 1992.

20.Øien G. E., Lepsø S. and Ramstad T. A., "An inner product space approach to image coding by contractive transformations", *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp.2773-2776, 1991.

21.Lepsø S. , Øien G. E. and Ramstad T. A., "Attractor Image Compression With Fast Non-iterative Decoding Algorithm", *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. V: 337-340, 1993.

22.Liu J., Marlow S. and Murphy N., "Multi-level fractal block coding in video compression", Proceedings of "1993 DSP - The Enabling Technology for Communication", Section 6.4. pp. 1-9, 1993.

23.Liu J., Marlow S. and Murphy N., "A new library search strategy for fractal block coding techniques", Proceedings of Irish Colloquium on DSP & Control, pp. 180-187, June 1993.

24.Fisher . Y., "Fractal Image Compression", SIGGRAPH'92 Course Notes, 1992.

25.Fisher Y. and Lawrence, "Fractal image encoding: SBIR phase I final report", tech. rep., NETROLOGIC, Inc., 1990.

26.Fisher Y., Jocobs E. W., and Boss R. D., "Fractal image compression using iterated

transforms", In J. Storer (ed.): *Image and Text Compression,* Kluwer Academic Publishers, Norwell MA, pp. 35-61, 1992.

27.Thomas L. and Deravi F., "Pruning of the transform space in block-based fractal image compression", *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. V: 341-344, 1993.

28.Kreyszig E., *Introductory Functional Analysis with Applications*, John Wiley & Sons, New York 1978.

29.Luenberger D. G. *Optimization by Vector Space Methods*. John Wiley & Sons, 1969.

30.Huffman D. A.,"A method for the construction of minimum redundancy codes", *Proc. IRE,* Vol. 40, Sept. 1952, pp. 1098-1101.

31.Huang T. S.,"PCM picture transmission",*IEEE Spectrum,* vol. 12, Dec. 1965, pp. 57-63.

32.Netravali A. N., "On quantizers for DPCM coding of picture signals", *IEEE Trans. Inform. Theory,* Vol. IT-23, May 1977, pp. 360-370.

33.Ahmed N., Natarajan T., and Rao K. r., "Discrete cosine transform", *IEEE Trans.*

Computers, Vol. c-23, Jan. 1974, pp. 90-93.

34.Chen W. H., smith Smith C. H., and Fralick S. C., "A fast algorithm for the discrete cosine transform", *IEEE Trans. On communications*, Vol. COM-25, No.9, pp.1004-9, September 1977.

35.Hou H. S., "A fast recursive algorithm for computing the discrete cosine transform", *Proc. IEEE Int. Conf. On Acoustics, Speeach and Signal Processing*, October 1987.

36.Chen W. H. and Pratt W. K., "Scene adaptive coder", *IEEE Trans. Commun.,* Vol. COM-32, pp. 225-232, March 1984.

37.Wallace G. K., "The JPEG still-picture compression standard", *Comm. of ACM*, pp. 20-44, Apr. 1991.

38.Zeng B. and Venetsanopoulos a., "A JPEG-based interpolative image coding scheme", *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. V: 393-396, 1993.

39.Gray R. M., "Vector quantization", *IEEE ASSP Mag,* April 1984, pp. 4-29.

40.Linde Y., Buzo A., and Gray R. M., "An algorithm for vector quantizer design",

IEEE Trans. Commun., Vol. COM-28, pp. 84-95, Jan. 1980.

41.Seferidis V., "Three dimensional block matching motion estimation", Electronic Letters, 27th August, 1992 Vol.28 No.18.

42.Bierling M., "Displacement estimation by hierarchical blockmatching", SPIE Vol. 1001 *Visual Communications and Image Processing'1988.*

43.Li H. and Forchheimer R., "A new motion-compensated technique for video compression", *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. V: 441-444, 1993.

44.Mussmann H. G., Hotter M. and Ostermann J., "Object-oriented analysis-synthesis coding of moving images", Signal Processing: *Image communication,* 1 pp. 117-138, 1989.

45.Hotter M., "Object-oriented analysis-synthesis coding based on moving two-dimensional objects", Signal Processing: *Image Communication*, 2 pp. 409-428, 1990.

46.Aizawa K., Harashima H. and Siato T., "Model-based synthesis image coding system", *Picture Coding Symp.*, Pres. No. 3.11, Stockholm, Sweden, June 9-11, 1987.

47.Adiv G., "Determining three-dimensional motion and structure from optical flow generated by several moving objects", *IEEE Pattern Anal. Mach. Intell.*, Vol. PAMI-7, No. 4, pp.384-401, July 1985.

48.Busch H., "Subdividing non rigid 3D objects into quasi rigid parts", *IEE 3rd Int. Conf. Image Process. Applic.*, Warwick, U.K., July 1989.

49.Eckman P. and Friesen V.W., *Facial Coding System,* Consulting Psychologists Press Inc, 577 College Arc, Palo Alto, 1977.

50.Eden M. and Kocher M., "On the performance of a contour coding algorithm in the context of image codng - Part I: Contour segment coding", *Signal Process.*, Vol. 8, No. 4, pp.281-386, July 1985.

51.Forchheimer R. and Fahlander O., "Low bit rate coding through animation", *Picture Coding Symp.*, Pres. No. 13.5, Davis, CA, March 1983.

52.Forchheimer R., Fahlander O. and Kronander T., "A sematic approach to the transmission of face images", *Picture Coding Symp.*, Pres. No. 10.5, Cesson-sevigne, July, 1984.

53.Geuen W. and Kappei F., "Principle strategy of model based source coding", *Picture*

Coding Symp., Pres. No. 12.2, Stockholm, Sweden, June, 1987.

54.Hotter M. and Thoma R., "Image segmentation based on object oriented mapping parameter estimation", *Signal Process.*, Vol. 15, No. 3, pp. 315-334, October 1988.

55.Kanako M., Koike A. and Hatori Y., "Codings with knowledge-based analysis of motion pictures", *Picture Coding Symp.*, Pres. No. 12.3, Stockholm, Sweden, June 9-11, 1987.

56.Welsh W. J., "Model-based coding of moving images at very low bit rates", *Picture coding symp.*, Pres. No. 3.9, Stockholm, Sweden, June 9-11, 1987.

60.Liou M. L, "Visual Telephony as an ISDN Application", *IEEE Commun. Mag*, Feburary 1990.

61.Habibi A. ed., Special issue on Image Bandwidth Compression, *IEEE Trans. on Communications, Vol.* COM-25, November, 1977.

62.Habibi A., "Survey of adaptive image coding techniques", *IEEE Trans. Commun., Vol.* COM-25, Mpv/ 1977, pp. 1275-1284.

63.Limb J. O., Rubinstein C. B., Thompson J. E., "Digital coding of color video signals

- A Review", IEEE Transactions on Communications, Vol. COM - 25, No. 11, November 1977.

64.Kretz f. and Nasse D., "Digital television: transmission and coding", *Proc. IEEE,* Vol. 73, pp. 575-591, April 1985.

65.Limb J. O., Rubinstein C. B., and Thomson J. E., "Digital coding of color video signals - A review", *IEEE Trans. Commun.,* Vol. COM-25, pp. 1349-1385, Nov. 1977.

66.Netravali A. N. and Haskell B. G., *Digital Pictures: Representation and Compression.* New York: Plenum Press, 1988.

67.Netravali A. N. and Limb L. O., "Picture coding: A review", *Proc. IEEE,* Vol. 68, pp. 366-406, March 1980.

68.Reeve H. C. III and Lim J. S., "Reduction of blocking effects in image coding", *J. of Opt. Eng.,* Vol. 23, pp. 34-37, Jan./Feb. 1984.

69.Sabri S. and Prasada B., "Video conferencing Systems", *Proc. IEEE,* Vol. 73, April 1985.

70.Schreiber W. F., "Picture coding", *Proc. IEEE,* Vol. 55, pp. 320-330, March 1967.

71.Shannon C. E., "A mathematical theory of communication Parts I and II", *Bell syst. tech J.,* Vol. 27, pp. 379-423; pp. 623-656, July 1948.

72.Woods J. W. and O'Neil s. d., "Subband coding of images", *Proc. IEEE Int. Conf. On Acoustics, Speeach and Signal Processing,* Vol. ASSP-34, pp. 1278-1288, Oct. 1986.

73.JPEG digital compression and coding of continuous-tone still images, Draft ISO 10918, 1991.

74.Wallace G. K., "The JPEG still picture compression standard", Comm. ACM, Vol. 34, No.4, pp.30-34, 1991.

75.CCITT Recommendation H. 261, " Video codec for Audiovisual Services at px64 kbits/s"

76.Standard Draft, Mpeg Video Committee Draft, MPEG 90/176 rev. 2, Dec. 1990.

77.Video codec for audio visual services at p x 64 kb/s, CCITT Recommendation H.261, 1990.

78.Coding of moving pictures and associated audio, Committee Draft of standard ISO 11172: ISO/MPEG/90/176, Dec, 1990.

79.Description of reference model 8 (RM8), CCITT Specialist Group on visual Telephony, June 1989.

80.Musmann H. G., Pirsch P., and Grallert H. J., "Advances in picture coding", Proc. of the IEEE, Vol. 73, pp. 523-548, Apr. 1985.

81.Jain A. K., "Image Data Compression: A Review", Proceedings of the IEEE, Vol. 69, No. 3, March 1981.

82.Jain A. K., "Displacement measurement and its application in interframe image coding", *IEEE Trans. Commun.,* Vol. COM-29, pp. 1799-1808, Dec. 1981.

83.Abramson A, ed., *Information Theory and Coding.* New York: McGraw-Hill, 1983.

84.Gallagher R. G., *Information Theory and Reliable communication.* New York: wiley, 1968.

85.Lim Jae S., *Two-Dimensional Signal and Image Processing*, Prentice-Hall 1990

86.Clarke R. J. *Transform Coding of Images*. London: Academic Press, 1985

87.Jahne Bernd, Digital Image Processing Concepts, Algorithms and Scientific Applications, Springer-Verlag Berlin, Heidelberg 1991.

88.Held Gilber, *Data Compression Techniques and Applications Hardware and Software considerations*, John Wiley & Sons, New York.

89.Jain Anil K., *Fundamentals of Digital Image Processing*, Englewood Cliffs, Prentice Hall, , 1989.

90.Jayant N. S. and Noll P., *Digital Coding of Waveforms,* Englewood cliffs, Prentice-Hall, 1984.

91.Pratt W. K., *Image Transmission Techniques.* New York: Academic Press, 1979.

92.Rosenfeld A., ed.,*Multiresolution Image Processing and Analysis.* Berlin: Springer Verlag, 1984.

93.Peitgen, Ho.-O., and Saupe, D., (eds.), *The Science of Fractal Images,* Springer-Verlag, New York, 1988.

94.McGuire M., *An Eye for Fractals,* Addison-Wesley, Redwood City, 1991.

95.Peigten H-O. and Richter P.H., *The Beauty of Fractals,* Springer-Verlag, Berlin, 1986.

96.Peitgen H-O., Jurgens H., and Saupe D., *Fractals for the Classroom,* Springer-Verlag, 1991.

97.Barnsley M. F., *The Destop Fractal Design Handbook*, Academic Press, Inc., Boston, 1989.

98.Wegner T. and Peterson M., *Fractal Creations*, Waite Group Press, 1990.

Appendix

APPENDIX A:

DISTORTION AND FIDELITY METRICS

The following distortion and fidelity metrics are defined in terms of the discrete functions $f(x,y)$ and $g(x,y)$ that represent the original and reconstructed m x n images respectively. They are defined for integer values of x and y where $0 \leq x < m$ and $0 \leq y < n$.

Distortion metrics

· Mean-Square Error:

$$MSE = \frac{1}{mn}\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}(f(x,y)-g(x,y))^2$$

· Normalized Mean-Square Error:

$$NMSE = \frac{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}(f(x,y)-g(x,y))^2}{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}(f(x,y))^2}$$

· Root Mean-Square Error:

$$RMS = \sqrt{MSE}$$

· Peak Mean-Square Error:

$$PMSE = \frac{MSE}{A^2}$$

where A = *maximum possible value of* $f(x,y)$.

Fidelity metrics

- MSE-based Signal-to-Noise Ratio:

$$SNR_{mse} = \frac{1}{NMSE}$$

- *SNR expressed* in decibels:

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

- RMS-based Signal-to-Noise Ratio:

$$SNR_{rms} = \sqrt{SNR_{mse}}$$

- Peak Signal-to-Noise Ratio:

$$PSNR_{mse} = \frac{1}{PMSE}$$