IMPROVED FRACTAL IMAGE COMPRESSION:

CENTERED BFT WITH QUADTREES

By

TUNG-MING KOO

Bachelor of Science
National Chung-Hsing University
Taiwan, R. O. C.
1983

Master of Science
University of Massachusetts
at Lowell
Lowell, Massachusetts
1988

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 1995

IMPROVED FRACTAL IMAGE COMPRESSION:

CENTERED BFT WITH QUADTREES

Thesis Approved:

J Chandler
Thesis Advisor

D. E. Hedman

Kenneth E Case

Don Young

Thomas C. Collins
Dean of the Graduate College

ACKNOWLEDGMENTS

First I would like to express my sincere apreciation to my advisor, Dr. J. P. Chandler, for his support and intuitive suggestions. I would also like to thank my advisory committee members, Dr. K. M. George, Dr. G. E. Hedrick, and Dr. Kenneth E. Case for their thoughtful comments on my dissertation.

I am also grateful for the teaching assistantship provided by the Computer Science Department. Without it, I and my wife could not finish our degrees. I would like also to thank Dr. Huizhu Lu for providing me a campus job in the first year of my graduate studies.

I want to thank my parents, Tseng-Lu Ku and Hsiu-Yuin Pan, for their support in whatever endeavor I have decided to undertake. I also want to thank my sister, Meihua Koo, and her husband, Hsuan-Tsung Hsieh, for countless lunches, dinners and rescues for many of my troubles. I wish both of you luck on your Ph.D. oral defenses.

Finally and most importantly, my deepest love and thanks go to my wife, Huey-Yeh Lin, for her inspiration and endless support, and my son, Oliver Koo, who is the fountain of my joy.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

NOMENCLATURE

$f(x)$	the signal to be compressed
ψ	mother wavelet
$\{\psi_i\}$	basis functions
$f(x, y)$	the image to be compressed
z	the graylevel of a pixel at (x, y), $z = f(x, y)$
$\boldsymbol{X}$	some space
$\boldsymbol{R}$	the real number space
$\boldsymbol{R}^2$	the space of a rectangle
$\boldsymbol{R}^3$	the space of a grayscale image
P_i	a point
$\delta(P_1, P_2)$	the distance between P_1 and P_2
w	the contractive mapping of IFS
α	the contractivity of the contractive mapping
x^*	a fixed point in $\boldsymbol{X}$ space
sup	the supremum metric
B	an element in $\boldsymbol{X}$ space
B'	a transformed element in $\boldsymbol{X}$ space

W	the collage of w_i transformations
A	the attractor image
M_i	the scale factor of the support
θ_i	rotation angle of the support
S_{xi}, S_{yi}	displacement in x and y direction of the support
D_i	the support of the domain block
R_i	the support of the range block
v_i	the contractive mapping of PIFS
α_i	the scaling factor of the contractive mapping
β_i	the offset factor of the contractive mapping
V	the collage of v_i transformations
f_i	the decompressed image sequence where $i = 0 .. \infty$
d_i	the image surface above D_i
r_i	the image surface above R_i
SE	square error between two image blocks
RMS	root mean square error
$\bar{r}_i$	a flat image block with graylevel equal to the mean of the range block r_i
$\bar{d}_i$	a flat image block with graylevel equal to the mean of the domain block d_i
β_i'	the new βi for centering method, $\beta_i' = \bar{r}_i$
a_i	the coefficient of x^i term
b_j	the coefficient of y^j term
c_{ij}	the coefficient of $x^i y^j$ term

$\tilde{\beta}_i$ the new β_i for re-centering method

CHAPTER I

INTRODUCTION

"One picture is worth a thousand words." This is one reason that images are massively used in our modern society. The massive usage of images causes difficulty of storage and transmission. For example, a grey-level image of about 1/4 screen of an ordinary SVGA monitor is composed of 512 pixels in width and 512 pixels in length. There are totally 262,144 pixels. Suppose the intensity of each pixel is from 0 to 127; that takes 8 bits (1 byte) to express. That image takes 262,144 bytes to store. For a full color image, which uses one byte for each R/G/B color on every pixel, of the same resolution and size, 768,432 bytes are needed. To transmit that full color image by a 9600 bps modem, it takes 655 seconds plus some time for the transmission overhead. Hence, image compression techniques are developed to shorten the transmission time and reduce the storage burden. In this study, we seek to improve the state-of-the-art image compression technique in compression ratio, compression speed, decompression speed, and/or decompressed image fidelity.

1.1 Background Information

While the price of computer storage devices is dropping drastically and the data transmission speed is being enhanced rapidly, to store more data in less memory and transmit more data in less time are always desired. Data compression techniques provide some choices to fulfill this purpose. Sometimes, the data are very important so that the decompressed data must be exactly the same as the original data. The techniques that

satisfy this requirement are called lossless data compression. Sometimes, the data are not very critical so imprecise decompressed data are acceptable. The techniques that can provide this not-so-exact restoration are called lossy data compression.

1.1.1 Lossless compression

Lossless data compression techniques have been well developed and are approaching the theoretical limit [Stor88][Bell90]. The most common methods are Huffman coding, LZ77, LZ78, and Arithmetic coding. They are mostly applied to textual data and some binary data such as executable codes. For such kinds of data, lossless compression typically can save 8% to 76% of the storage space depending on the information entropy of the original data and the technique used [Burt95]. However, lossless compression techniques may expand (poor) or save 50% (good) storage space for most natural image data and voice data [Nels91] [Stor92]. The advantage of these techniques is that the decompressed data will be exactly the same as the original one, no matter how many compression-decompression processes are applied on the data. The disadvantage is that only a limited amount of space can be saved by these techniques.

1.1.2 Lossy compression

Most image and voice data are not critical in detail. For example, human eyes may not tell the difference if the intensity (graylevel) of one pixel of an image is changed to one level brighter. In contrast, it may be a disaster to change the value of one bit of an executable file. Minor noise embedded in the data may cause the degradation of the visual or acoustic quality for the image data or voice data respectively, but the information

contents could still be sensible. If the noise is very weak, human eyes or ears might not even perceive the existence of the noise. That is the reason why lossy compression is mainly applied to image and voice data.

There are several lossy compression techniques very popular such as vector quantization and JPEG for still image data, fractal and wavelet compression for still image and video data, and MPEG for video data. Fractal image compression is one of the most promising methods. The theory behind fractal compression is also applicable to one-dimensional data such as voice data [Vine93]. Barnsley had demonstrated a compression rate of as much as 10,000:1 [SciAm88], although this image was not naturally occurring, nor is the compressing automatic.

Comparing with lossless compression, the advantage of lossy compression is that more storage space can be saved. The disadvantage is the degradation of decompressed data, and this degradation exists for each decompression if you do multiple compression-decompression cycles on one image.

1.1.2.1 Vector Quantization (VQ)

The Vector Quantization idea comes from Scalar Quantization by which all values can be represented with fixed subset of representative values. For example, 16-bit values can be represented by only the 8 most significant bits (MSBs) and result in an approximation of the original data at the cost of precision. In this case, the fixed subset contains all the 16-bit numbers divisible by 256, that is, 0, 256, 512, ..., etc. Hence, all quantization methods are lossy.

contains all the 16-bit numbers divisible by 256, that is, 0, 256, 512, ..., etc. Hence, all quantization methods are lossy.

As an extension of Scalar Quantization, Vector Quantization represents a small arrays of values instead of individual values [Zats95]. The color map is a typical VQ example. A full color image can be represented by a 2-dimensional array of triplets (RGB values). For most natural images, the whole RGB space is not fully used, i.e., the color tends to concentrate in certain areas. For example, the picture of a forest will typically have a lot of green. So, a relatively small subset of colors (e.g., 256 colors) can be the representatives. All RGB triplets then can be approximated by those representatives, so each color pixel can be represented by one byte instead of three.

To the VQ community, a "vector" is a small rectangular block of pixels. Similar to the color map example in which some colors occur much more frequently than others, some patterns occur much more frequently than others in an image. So the smart way is to store only a few of these common patterns in a separate file called the codebook. Some codebook vectors are flat, some are sloping, some contain tight texture, some sharp edges, etc. Each codebook entry is assigned an index number. A given image is partitioned into a regular grid array. Each grid element is represented by an index into the codebook.

1.1.2.2 The JPEG Standard

JPEG (Joint Photographic Experts Group) is a standard image compression mechanism. It is designed to compress natural real-world scenes instead of cartoons and line drawings. JPEG does not handle black-and-white images and motion pictures (video). JBIG and MPEG committees are working on those types of images respectively.

The JPEG algorithm achieves much of its compression by exploiting human eye limitations that small color details are not perceived as well as small details of brightness. The outline of the baseline compression algorithm for grayscale images is listed here:

1. Group the pixel values into 8×8 blocks. Transform each 8×8 block through a discrete cosine transform (DCT) which is similar to Fourier transform and give a frequency map. Thus, the average value and higher-frequency changes within a block are represented by some numbers. Some of the high frequency information can be discarded depending on what image fidelity the users want.
2. In each block, divide each frequency component by a separate quantization coefficient, and round the results to integers.
3. Encode the quantized coefficients using either Huffman or arithmetic coding.

JPEG also provides three optional modes: progressive mode, hierarchical mode, and lossless mode. Progressive mode is intended to support real-time transmission of images It allows the DCT coefficients to be sent incrementally in multiple scans of the image. Hierarchical mode represents an image at multiple resolutions. Lossless mode does not use DCT, since roundoff errors prevent a DCT calculation from being lossless. The lossless mode simply codes the difference between each pixel and the "predicted" value for the pixel. The predicted value is a simple predictor function, such as average, of the already-transmitted pixels just above and to the left of the current one [Wall91] [Nels91] [Penn93] [Lane95].

1.1.2.3 MPEG Standards

MPEG (Moving Picture Experts Group) works on standards for the compression of video and associated audio data. They have completed the first phase of three: MPEG-1, MPEG-2 and MPEG-4. MPEG-1 is defined as "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s." MPEG-2 is defined as "Generic Coding of Moving Pictures and Associated Audio." The first step of the MPEG-1 algorithm is to convert the color image into YUV space in which the Y channel should preserve better image fidelity when decoded (i.e., not much compression) and UV channels can be greatly compressed. The basic scheme is to predict motion from frame to frame in the temporal direction, and then to use DCTs to organize the redundancy in the spatial directions [Gall91] [AdPo94].

1.1.2.4 Wavelet theory

One of the most commonly used approaches for analyzing a signal $f(x)$ is to represent it as a weighted sum of simple building blocks, called basis functions:

$$f(x)=\sum_i c_i \psi_i(x) \tag{1.1}$$

where the $\psi_i(x)$ are basis functions and the c_i are coefficients, or weights. Since the basis functions ψ_i are fixed, it is the coefficients that contain the information about the signal.

The simplest such representation uses translations of the impulse function as its only bases, yielding a representation that reveals information only about the time domain behavior of the signal. Choosing the sinusoids as the basis functions yields a Fourier representation that reveals information only about the signal's frequency domain behavior.

Fortunately, low frequency events are spread out (or non-local) in time, and high frequency events are concentrated (or localized) in time. In order to get useful time-frequency information about a signal, the basis functions should be designed to act like cascaded octave bandpass filters, which repeatedly split the signal's bandwidth in half.

In wavelet compression, all basis functions are constrained in $\{\psi_i\}$ to be scaled and translated versions of the same prototype function ψ, known as the *mother wavelet*. The scaling is accomplished by multiplying x by some scale factor; if the scale factor is a power of 2, yielding $\psi(2^v x)$ where v is some integer, the cascaded octave bandpass filter structure is obtained. Because ψ has finite support (support is described in section 3.2), it will need to be translated along the time axis in order to cover an entire signal. This translation is accomplished by considering all the integral shifts of ψ,

$$\psi(2^v x - k), \qquad k \in Z \tag{1.2}$$

Putting this all together produces a wavelet decomposition of the signal,

$$f(x) = \sum_v \sum_k c_{vk} \psi_{vk}(x) \tag{1.3}$$

where $\psi_{vk}(x) = 2^{v/2} \psi(2^v x - k)$.

The $2^{v/2}$ is needed to make the bases orthonormal.

As to the coefficients c_{vk}, they are computed by the wavelet transform, which is just the inner product of the signal $f(x)$ with the basis functions $\psi_{vk}(x)$ [HiJaSe94].

1.1.2.5 Fractal Image Compression

The fractal image compression method is based on Iterated Functions Systems (IFS), the Collage Theorem, and the Contraction Theorem. The mathematics of IFS is

presented in *Fractals Everywhere* by Michael Barnsley [Barn89]. The Collage Theorem and the Contraction Theorem are also proven in that book. Heinz-Otto Peitgen compares IFS to a Multiple Reduction Copying Machine (MRCM) [PeJüSa93]. A MRCM is a regular copy machine with special features:

1. Multiple lenses.
2. Each lens creates its own reduced copy of the original.
3. Reduced copies overlap to each other.
4. The initial input copy can be anything.
5. The output copy is always fed back as new input.

When the MRCM runs, it eventually will produce an unique image which is called the attractor or the fixed point of that MRCM, because each lens creates its own reduced copy of the original. A good MRCM example that produces the Sierpinski triangle is presented in section 3.8 and Figure 3.5. Different numbers of lenses and lens arrangements will result in different attractors no matter what is the initial input image, as guaranteed by the Contraction Theorem which is stated in section 3.3 and 3.4.

Similarly, IFS is a set of contractive transformations that map from a defined rectangle to smaller portions of that rectangle. The production of IFS is called fractal image because of the self-similarity property. A fractal image can be represented by a small number of IFS transformation, hence the storage space is saved. The construction of a typical fractal image, the Sierpinski triangle, is shown in section 3.8.

Unfortunately, natural images are not self-similar, because a small portion of the image is hardly like the whole image. To avoid this problem, Barnsley's Ph.D. student Arnaud Jacquin in Georgia Institute of Technology proposed a Partitioned IFS (PIFS)

which does not map from the whole image to the parts, but from larger parts (domains) to smaller parts (ranges). Because the mappings of PIFS are still contractive, when iterated it will converge to its latent fixed point image. Constructing a PIFS amounts to pairing each range block to the domain block that it most closely resembles under some to-be-determined affine transformation. Affine transformations act to translate, scale, shear, and rotate points in the domains. A simple compression example based on the PIFS model is presented in section 4.5.

The decompression process begins with any initial image. Then the set of transformations is repeatedly applied on that image. The attractor will be stable after several iterations. The attractor usually will not be exactly identical to the original image [Komi94] [Komi95].

1.2 The Problems

The compression process is basically the pairing of each range block to a domain block such that the difference between the two, under an affine transformation, is minimal. A general description of finding a good PIFS for any given image involves five main issues [Komi94] [Komi95]:

1. Partitioning the image into range blocks.
2. Forming the set of domain blocks.
3. Choosing the types of transformations that will be considered.
4. Selecting a distance metric between blocks.
5. Specifying a method for pairing range blocks to domain blocks.

The range blocks and domain blocks are not necessarily squares or rectangles. Fisher et al. proposed an HV partitioning in which a range block is horizontally or vertically cut into two pieces alternatively when the pairing process fails [FiMe94]. Davoine et al. developed a triangular partitioning method and expected a lot of similarities between those triangles [DaCh94]. Both methods share the same problem that the locations (coordinates) of domain blocks and range blocks must be explicitly recorded in the compressed file, which decreases the amount of compression.

Fisher utilized the beauty of quadtree structures to develop *fractal image compression with quadtrees* in which the range locations are implicitly encoded with only one bit for each quadtree branch [Fish95]. However, the problem is that many more range blocks are generated by the quadtree partitioning, because each partition produces four sub-range blocks. Each produced sub-block will repeat the pairing process all over again and may be partitioned repeatedly if its pairing fails again. It is the pairing processes that make fractal image compression very slow. Besides, each successfully paired sub-block needs to record all its transformation coefficients in the compressed file, which worsens the compression.

Most researchers use the same type of transformation that Jacquin and Fisher used. That transformation, a scaling and a shifting on the domain pixel brightness, is so primitive that only a very similar domain block can be successfully paired to a range block. The transformaion is introduced in section 4.2. It makes the successful rate of pairing very low and lengthens the searching time.

Monro et al. from University of Bath, England, proposed a more effectual transformation called the Bath Fractal Transformation (BFT) [Monr93] [MoDu92a]

[MoDu92b] [MoNi94] [MoWo94]. Jacquin and Fisher's transformation is just a special case of the BFT. This transformation is so powerful that almost any domain block can be transformed into an acceptable block which barely approximates the range block. Monro declared that BFT image compression can do quite well without searching. That means the compression speed could be very fast with the cost of worse decompressed image fidelity. The image fidelity can be improved by simply allowing searching. However, the pairing time could be longer than Barnsley's because the transformation is more complicated and takes longer to calculate the coefficients and errors. Section 5.5 describes BFT in detail.

The decompression algorithm is quite straight forward. Applying all transformations on the initial image is called one decompression iteration. It usually takes 7 to 9 iterations to converge to the attractor image. This could be a problem for some applications in which a real-time image display is critical, such as video decompression.

1.3 Motivation

For lossy image compression, it is well known that there is no best method for all kinds of images, because the results always depend on the input data. A method may do very well on an image but very poorly on another. As to the word "best," it is also ambiguous: best compression ratio, best compression speed, best decompressed quality, or best decompressing speed. These four intentions have trade-offs among them.

Under such circumstance, a good method means to achieve more on some of these four intentions with less cost of others.

1.4 Objectives of the Study

In this study, we propose a better fractal image compression model with two innovations: *centering* and *BFT with quadtrees*. We expect that centering can reduce the decompression time to about half of the traditional models' time and the contents of the decompressed image could be perceptible as early as the end of the first decompression iteration. We also expect this early convergence property could make the result of the second decompression iteration very similar to the attractor image. Thus, the centering method provides a good choice for real-time video decompression.

The *BFT with quadtrees* is a modified algorithm from Fisher's *fractal image compression with quadtrees* [Fish95]. The goals of developing this method are:

1. To compress an image into smaller size.
2. To reduce the decompression time.
3. To keep the decompressed quality at the same level.

We expect this modified algorithm is not only useful for quadtree type partitioning models, but also applicable to all kinds of partitioning models such as HV and triangular partitions.

1.5 Outline of the Dissertation

The remainder of this dissertation is organized as follows. Chapter II provides the literature review for fractal image compression. Chapter III describes the mathematical foundation of fractal image compression in detail, including the contractive transformations, the fixed point theorem, the iterated function systems, and the collage theorem. Chapter IV introduces Jacquin's partitioned iterated function systems and our

centering modification. Preliminary results of the centering method are also presented in this chapter. Chapter V shows the algorithms of Fisher's quadtree-based fractal encoding scheme, Monro's Bath Fractal Transformation, and our centered BFT with quadtrees. Chapter VI presents the experimental results from the centering and our centered BFT with quadtrees, and compares them with the results of Fisher's original model. Chapter VII summarizes our study and the prospects for future work.

CHAPTER II

THE LITERATURE REVIEW

2.1 Introduction

While the root of fractal image compression is the Iterated Function System (IFS) which is based on the work of Williams [Will71] and Hutchinson [Hutc81], the rebirth of fractal geometry is usually traced to IBM mathematician Benoit B. Mandelbrot and his 1977 seminal publication *The Fractal Geometry of Nature* [Mand77]. This book put forth a powerful thesis: traditional geometry with its straight lines and smooth surfaces does not resemble the geometry of trees and clouds and mountains. Fractal geometry, with its convoluted coastlines and endless detail, does [Mand77]. This insight opened vast possibilities, so that IFS's became very popular in the mid 1980's.

2.2 Barnsley's IFS

In 1985, Barnsley and his coworkers at Georgia Institute of Technology focused on modeling natural shapes such as leaves and clouds. In the reverse thinking direction of generating synthesized image, Barnsley et al. realized that fractal mathematics should be good for representing images. They announced the incredible compression rates of over 10,000 to 1 in Popular Science magazine [SciAm88] [BaSl87][BaSl88]. The algorithm first partitioned an image into self-similar parts with a human's interactive help. Then each part was coded as an IFS under the criterion of the Collage Theorem. This method is derisively referred to as the "graduate student algorithm": locking up a graduate student in

a tiny office with a workstation and not letting him out until they come up with a good IFS for your image. They were granted two patents [BaSl90] [BaSl91] and established a new company, Iterated Systems, Incorporated.

The myth of 10,000 to 1 compression rate depended on the fact that the demonstrated images are all man-made. Not only the images are synthesized, but also this deadly slow process is not automatic. In 1988, Barnsley admitted: "Complex color images require about 100 hours each to encode and 30 minutes to decode on the MassComp dual processor workstation." That is 100 hours of human-guided computation.

After Barnsley published *Fractals Everywhere* [Barn89] [Barn93], he and Hurd published a second book, *Fractal Image Compression* [BaHu92]. In *Fractals Everywhere,* he presented the mathematics of IFS, and proved the Collage Theorem.

2.3 Jacquin's PIFS

In 1989, Jacquin, a Barnsley's Ph.D. student, proposed a fully automated algorithm, Partitioned Iterated Function Systems (PIFS), for fractal image compression based on Barnsley's IFS. The algorithm is described in Jacquin's landmark paper "Image Coding Based on a Fractal Theory of Iterated contractive Image Transformations," and all contemporary fractal image compression algorithms are based upon this paper [Jacq92]. A grey-scale image is partitioned into nonoverlapping square range blocks. The same image is partitioned again into some overlapping larger square domain blocks sorted into three categories (shade blocks, edge blocks and midrange blocks). Each range block searches a domain block of the same category in order to find a domain block that minimizes the "distance" between the range block and the affine transformed domain

block. The affine mapping first shrinks the domain block down to the size of the range block, then scales its grey-levels and adds to a constant grey-tone block. There are several different definitions of the distance between two blocks such as Euclidean metric and supremum metric. The supremum metric is defined in section 3.5.

Jacquin's algorithm is simple and not speedy, but it is fully automatic with the price of a much worse compression rate (compared with Barnsley's 10,000 to 1). His work provides a starting point for further research in many directions:

1. The partitioning of the image into ranges: adaptive quadtrees, HV rectangular and triangular ranges,
2. The encoding: choice of the domain pool, including several fixed basis blocks and even several image domain blocks for the code of a range,
3. Classification methods for the complexity reduction of the encoding step: based on image values and intensity variance, clustering of domains, fast algorithms from computational geometry to solve nearest neighbor problems,
4. The decoding: standard iteration vs. fast hierarchical or direct numerical,
5. Coding of 1-D or 3-D data: time series, volume data, video frames.

2.4 Monro's BFT

In 1992, Monro and Dudbridge from the University of Bath, England, generalized Jacquin's scheme with the Bath Fractal Transform (BFT) [MoDu92a]. The domain block is mapped onto the range block by a four-variable BFT, while Jacquin uses a two-variable affine transformation. The four-variable BFT is illustrated in section 5.5. In order to minimize the error of the mapping, the least- squares error technique is applied. The four

variables can be obtained by solving a set of linear equations in linear time with the total number of pixels, thus leading to higher computational efficiency as compared to other techniques. Although the compression is faster, the compression-fidelity tradeoff of the primitive order-1 BFT is a little bit worse than that of JPEG [MoDu92b].

In 1993, Monro further generalized his previous work and implemented an order-3 BFT with different levels of local search. The encoding cost increases with both search level and order of the BFT, but the accuracy of the code is improved and beats JPEG coding. It also gives a lower Root-Mean-Square (RMS) error than Jacquin's PIFS-coding does with the same search level. Monro concluded that variants of the BFT will offer better fidelity-compression tradeoffs than JPEG coding and therefore are a serious alternative as a next-generation method for still-image coding [Monr93].

CHAPTER III

THE MATHEMATICAL FOUNDATION OF FRACTAL IMAGE COMPRESSION

3.1 Introduction

In this chapter, we describe the underlying mathematical principles of fractal image compression based on the theory of contractive iterated function systems. For a thorough introduction, the reader is referred to [Barn93].

We first define the mathematical model of an image. Then we describe the contractive transformations and their properties. Next, a short proof for the fixed point theorem is given. We define the supremum matrix and explain how the fixed point theorem can also be applied to images. The iterated function system is defined and is supported by the fixed point theorem. Based on the IFS, the collage theorem is derived. At the end of this chapter, we use the Sierpinski triangle image as an example to link IFS theory with image compression.

3.2 Definitions

In order to discuss the compression of digital images, we need a mathematical model of an image. An image is considered as an element in the space of functions of two variables which are defined on a rectangular support ($\boldsymbol{R}^2$). The two variables are denoted as x and y. Since the point (x, y) is defined on the support, (x, y) belongs to $\boldsymbol{R}^2$. A black-and-white image is a set of binary points on that support.

$$\{z \mid z = f(x, y),\ (x, y) \in \mathbb{R}^2,\ z = 0 \text{ or } 1\} \tag{3.1}$$

A black pixel is represented by $z = 1$, and a white pixel is represented by $z = 0$. A grayscale image can be defined as a function on a rectangular support.

$$\{z \mid z = f(x, y),\ (x, y) \in \mathbb{R}^2,\ z \in \mathbb{R}\}. \tag{3.2}$$

For example, the face of the photographer is cropped from the Camera image (Figure 3.1) which is one of our test images. The face image is on a support of the x-y plane. The dimension of x-y support is 16×16. The z value, from 0 to 255, represents the graylevel of a pixel at location (x, y). Figure 3.2 is the mathematical model of the face image in $\mathbb{R}^3$.

In the remainder of this dissertation, we will use the term image to mean a grayscale image which is denoted as the function $f(x, y)$.

3.3 Contractive Transformations

Since a grayscale image f is a function of (x, y) location, it can also be denoted as a triplet

$$(x, y, f(x, y)) \quad \text{where } (x, y) \in \mathbb{R}^2 \text{ and } f(x, y) \in \mathbb{R} \tag{3.3}$$

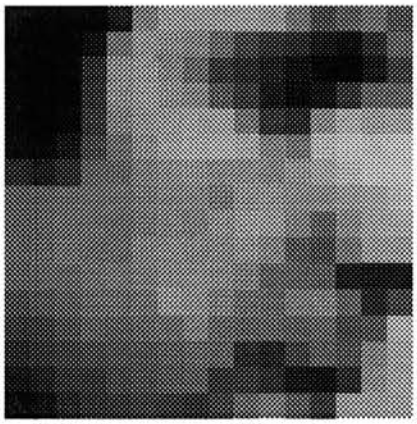
So, a grayscale image can be considered as a set of points in $\mathbb{R}^3$. Similarly, a black-and-white image can be denoted as a pair

$$(x, y) \qquad \text{where } (x, y) \in \mathbb{R}^2 \text{ and } f(x, y)=1. \tag{3.4}$$

So, a black-and-white image can be considered as a set of points in $\mathbb{R}^2$.

Now, let us generalize this: an image can be considered as a set of points in some space X. The most common choice of underlying space is that of $X = \mathbb{R}^n$. A black-and-

(a) Original Camera image



(b) The cropped face image

Figure 3.1 (a) Original Camera image and (b) the cropped face image.

white image can thus be considered as a subset of sample points in $\boldsymbol{X} = \boldsymbol{R}^2$. A grayscale image can then be seen as a subset of points in $\boldsymbol{X} = \boldsymbol{R}^3$, which represents a set of points on a surface in $\boldsymbol{R}^3$. This surface is the graph of the function representing the image intensity. Figure 3.2 shows the surface of the face image.

Between two elements $P_1 \in \boldsymbol{X}$ and $P_2 \in \boldsymbol{X}$, there are several possible metrics which are functions that measures distance, $\delta(P_1, P_2)$. The Euclidean metric in $\boldsymbol{R}^n$ is often used, although other choices such as supremum metric are also adopted. The supremum

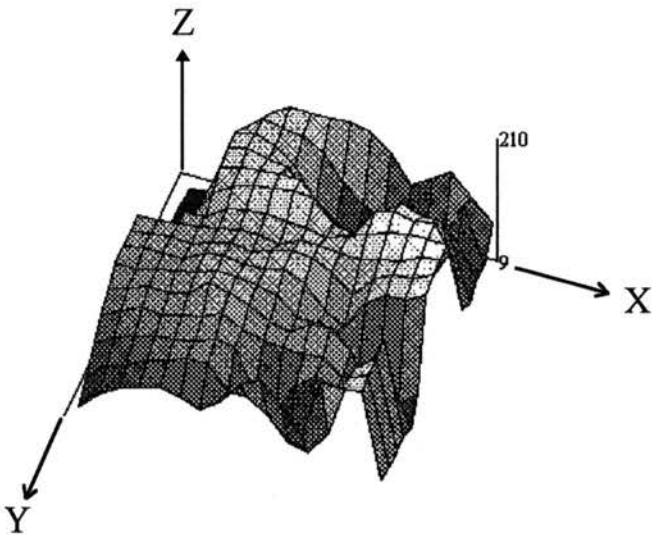


Figure 3.2 The 3-D mathematical model of the face image.

metric is defined in section 3.5. In fact, the choice of metric determines whether the transformations we use are contractive or not [Fish95]. A good example at the end of this section illustrates that a transformation might be contractive with some metrics but not contractive with other metrics.

Now, consider a mapping

$$w: X \mapsto X. \tag{3.5}$$

The space $\boldsymbol{X}$ can be the space of compact subsets of $\boldsymbol{R}^2$ for black-and-white images, or the space of compact subsets of $\boldsymbol{R}^3$ for grayscale images. Moreover, $\boldsymbol{X}$ could be chosen as the space of functions defined on a unit square, or the space of functions defined on the unit interval for one-dimensional signals. We will use a simple one-dimensional example to show a contractive mapping and the fixed point theorem in section 3.4. The contractive mapping means two points in $\boldsymbol{X}$ will become closer after each application of the mapping.

$$\delta(w(P_1), w(P_2)) \leq \alpha \cdot \delta(P_1, P_2), \qquad \forall_{P_1, P_2 \in X}, \tag{3.6}$$

whenever $0 \leq \alpha < 1$, the mapping w is said to be contractive with contractivity α [Nova93a].

Example

Let us consider a mapping $w: \boldsymbol{R}^2 \mapsto \boldsymbol{R}^2$,

$$w\begin{bmatrix} x \\ y \end{bmatrix} = 0.99\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}.$$

If the L_2 norm is used as the distance metric, this transformation is contractive with contractivity 0.99.

If the L_1 norm is used as the distance metric, this transformation is not contractive. For example,

$$w\begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0.99\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{0.99}{\sqrt{2}} \\ -\frac{0.99}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0.7 \\ -0.7 \end{bmatrix}.$$

The definition of L_1 is the summation of the absolute values of all dimensions, so

$$\left\| \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\|_1 = 1 \; < \; \left\| \begin{bmatrix} 0.7 \\ -0.7 \end{bmatrix} \right\|_1 = 1.4.$$

If the L_∞ norm is used as the distance metric, this transformation is not contractive. For example,

$$w\begin{bmatrix} 1 \\ 1 \end{bmatrix} = 0.99\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.99\sqrt{2} \\ 0 \end{bmatrix} = \begin{bmatrix} 1.4 \\ 0 \end{bmatrix}.$$

The definition of L_∞ is the maximum value of the absolute values of all dimensions, so

$$\left\| \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\|_\infty = 1 \; < \; \left\| \begin{bmatrix} 1.4 \\ 0 \end{bmatrix} \right\|_\infty = 1.4.$$

Hence, whether a transformation is contractive depends on the choice of the distance metric. Of course, if a transformation is contractive in one norm, then $\lim_{n\to\infty} \left\| w^{(n)}(x) \right\| = 0$ in every norm, but a particular iteration in one of these norms may not be contractive with an $\alpha < 1$.

3.4 The Fixed Point Theorem

The contractive transformation ensures that a fixed point, $x^* \in \boldsymbol{X}$, will be reached if we repeatedly apply the transformation w, such that

$$w(x^*) = x^*. \tag{3.7}$$

This theorem can be formally described as:

Theorem

let w be a contractive transformation, and

$$w^{(n)}(x) = \underbrace{w(w(w\cdots(x)))}_{n} = \underbrace{w \circ w \circ w \cdots}_{n}(x), \tag{3.8}$$

then for any $x \in \boldsymbol{X}$, $x^* \in \boldsymbol{X}$ will be the fixed point

$$\lim_{n\to\infty} w^{(n)}(x) = x^*. \tag{3.9}$$

Furthermore, if we are given a point $x_0 \in \boldsymbol{X}$ then

$$\delta(x^*, x_0) \le \frac{1}{1-\alpha} \delta(w(x_0), x_0). \tag{3.10}$$

This means that for a given x_0, if we find a mapping $w(\cdot)$ such that $\delta(w(x_0), x_0)$ is small, then the distance from x^* (the fixed point of $w(\cdot)$) to x_0 is also small.

Proof

Using the triangle inequality repeatedly,

$$\begin{aligned}\delta(x_0,x^*) &\le \delta(x_0,w(x_0))+\delta(w(x_0),w^{(2)}(x_0))+ \\ &\le \delta(x_0,w(x_0))\cdot(1+\alpha+\alpha^2+\cdots) \\ &\le \frac{1}{1-\alpha}\delta(x_0,w(x_0)).\end{aligned} \tag{3.11}$$

The equations (3.7), (3.9) and (3.10) are called the fixed point theorem.

If we wish to find a mapping with a fixed point as close as possible to a given point, we should seek for a contractive mapping that moves this given point as little as possible.

Example Let $\boldsymbol{X}$ be the space of a one-dimensional signal and let the Euclidean metric be used to measure the distance,

$$\delta(x, y)=|x-y|.$$

Consider a mapping $w: X \mapsto X$,

$$w(x)=0.5\cdot x+2.$$

It is easy to verify w is a contractive mapping, because for any $x, y \in \boldsymbol{X}$

$$\begin{aligned}\delta(w(x),w(y)) &= |0.5x+2-(0.5\cdot y+2)| \\ &= |0.5\cdot(x-y)| \\ &= 0.5\cdot|x-y| \\ &= 0.5\cdot\delta(x,y)\end{aligned}$$

That means, the distance between x and y becomes closer after a contractive mapping. The contractivity is $\alpha = 0.5$.

What is the fixed point x^* for the contractive mapping w? According to the definition of the fixed point theorem (Equation 3.7),

$$w(x^*) = x^*$$

$$\Rightarrow 0.5\,x^* + 2 = x^*$$

$$\Rightarrow x^* = 4.$$

Because $\boldsymbol{X} = \boldsymbol{R}$, the fixed point x^* is easy to calculate. However, the fixed point x^* is very difficult to calculate directly when $\boldsymbol{X}$ is a space of compact subsets of $\boldsymbol{R}^n$. Suppose the fixed point x^* is unknown, and the number which we want to represent by this contractive mapping w is 4.1, i.e., $x_0 = 4.1$, $x_0 \in \boldsymbol{X}$. What is the possible inaccuracy introduced by this representation?

$$\begin{aligned}\delta(x^*, x_0) \le \frac{1}{1-\alpha}\delta(w(x_0), x_0) &= \frac{1}{1-0.5}\delta(w(4.1), 4.1) \\ &= \frac{|w(4.1), 4.1|}{1-0.5} \\ &= \frac{|4.05 - 4.1|}{0.5} \\ &= 0.1\end{aligned}$$

That means, if we represent 4.1 with this contractive mapping w, the error in this representation will be less than or equal to 0.1. This explains the fixed point theorem: if a contractive mapping can only move a given point a little bit (i.e., a very small $\delta(w(x_0), x_0)$), then that given point will be very close to the fixed point x^* (i.e., a small $\delta(x^*, x_0)$).

3.5 Supremum metric

The contractive mapping maps an element of the space $\boldsymbol{X}$ to space $\boldsymbol{X}$, where $\boldsymbol{X}$ can be not only a space of $\boldsymbol{R}^n$, but also a space of compact subsets of $\boldsymbol{R}^n$. That is, the fixed point theorem is also good for a space of images.

Let $\boldsymbol{X}$ be a space of compact subsets of $\boldsymbol{R}^3$. The distance between two images, $f \in \boldsymbol{X}$ and $g \in \boldsymbol{X}$ can be defined by the supremum metric:

$$\delta(f,g) = \sup_{(x,y) \in R^2} |f(x,y) - g(x,y)|. \tag{3.12}$$

This metric finds the position *(x, y)* where two images f and g differ the most and sets this value as the distance between f and g.

3.6 Iterated Function Systems

An Iterated Function System (IFS) is a set of contractive mappings

$$\{w_i\}_{i=1}^{N} \tag{3.13}$$

where the contractivity of mapping w_i is α_i. Barnsley had described IFS in detail in [Barn93]. We briefly explain how contractive mappings compose the IFS as follows.

If we are given a set $B \subset \boldsymbol{X}$ then this set is mapped by w_i onto another set B' through

$$B' = \bigcup_{x \in B} w_i(x). \tag{3.14}$$

That means w_i transforms a set of elements to another set. Hence, we redefine the w_i as

$$w_i : \tilde{X} \mapsto \tilde{X}, \tag{3.15}$$

where the space $\tilde{X}$ is the space of all compact subsets of X.

Equation 3.14 can be rewritten as: $B \in \tilde{X}$ and $B' \in \tilde{X}$

$$B' = w_i(B). \tag{3.16}$$

A group of such contractive mappings, w_i, compose the IFS (defined in Equation 3.13). This group mapping is defined by

$$W : \tilde{X} \mapsto \tilde{X}, \tag{3.17}$$

and

$$W(B) = \bigcup_{i=1}^{N} w_i(B). \tag{3.18}$$

The mapping $W(\cdot)$ can be shown to be contractive with contractivity $\alpha = max_i\ \alpha_i$. Then it is said that the IFS has contractivity α.

An IFS has, analogously to a contractive mapping, a unique fixed point. By this we mean a set $A \in \tilde{X}$ which is invariant to $W(\cdot)$, such that

$$W(A) = A. \tag{3.19}$$

This fixed point is commonly referred to as the attractor of the IFS.

Let $B \in \tilde{X}$ and

$$W^{(n)}(B) = \underbrace{W(W(W \cdots (B)))}_{n}. \tag{3.20}$$

Then for any such B

$$\lim_{n \to \infty} W^{(n)}(B) = A. \tag{3.21}$$

Thus, if we start with any compact subset of $\boldsymbol{X}$ and apply W iteratively on the result from the previous step, we end up in the attractor of the IFS. This can be viewed as saying that the information about the attractor is stored in the mappings w_i of the IFS.

3.7 The Collage Theorem

Let A be the attractor of the IFS. From the relation

$$A = W(A) = \bigcup_{i=1}^{N} w_i(A), \tag{3.22}$$

it can be seen that A consists of the union of contractively mapped copies of itself. It is said that $W(A)$ is a *collage* covering A.

Theorem 3.1 (The Collage Theorem) Let f be an image (strictly, an element in $\tilde{\boldsymbol{X}}$). Let also $\delta(\cdot,\cdot)$ be a metric on $\tilde{\boldsymbol{X}}$. If we are given an IFS

$$\{w_i\}_{i=1}^{N} \tag{3.23}$$

with attractor A and a mapping $W(\cdot)$ deduced from the IFS member functions w_i, and if $W(\cdot)$ is contractive under $\delta(\cdot,\cdot)$ with contractivity α, then it follows that

$$\delta(A,f) \le \frac{1}{1-\alpha}\,\delta(W(f),f). \tag{3.24}$$

The collage theorem is analogous to the fixed point theorem in section 3.3 (Equation 3.10). For a proof, see [Barn93].

This inequality states that if we can find a set of mappings which produce a collage, sufficiently close to the original image, then the attractor of the corresponding IFS will also be 'close' to the image. This is an important statement, since it is a much easier

task to find a good collage from a set of mappings $\{w_i\}_{i=1}^{N}$ than to calculate an attractor for every different choice of mappings. Specifically when the member functions w_i of the IFS are contractive with contractivities α_i, then the mapping $W(\cdot)$ defined by

$$W(B)=\bigcup_{i=1}^{N} w_i(B) \tag{3.25}$$

is contractive with contractivity $\alpha = max_i\ \alpha_i$.

3.8 The Sierpinski Triangle Example

According to the right-hand side of Equation 3.25, the *whole* image, B, will be transformed into part of itself to be collaged. This is the deficiency of IFS theory, because natural images usually are not self-similar. However the theory does apply to some man-made images and provides a good foundation for further development.

In this section, we use the famous Sierpinski triangle, a man-made image, to show the application of IFS in image compression. Figure 3.3 is the Sierpinski triangle that has strong self-similarity.

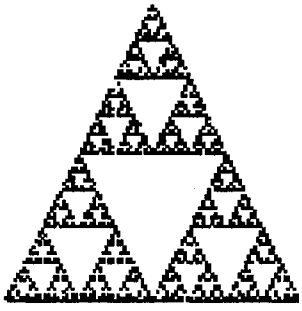


Figure 3.3 Sierpinski triangle

Our objectives are to compress this image and use the compression process to explain the meaning of those symbols and theorems defined in previous sections.

Similar to the surface of the face image (Figure 3.2), the space $\boldsymbol{X}$ of the Sierpinski triangle is $\boldsymbol{R}^3$. Every pixel is a triplet, $(x, y, f(x,y))$. We use f to represent the surface of the image.

Since we are dealing with an image, not a single point, we need a space which is a collection of all images. The $\tilde{\boldsymbol{X}}$ is defined as a space of all compact subsets of $\boldsymbol{X}$. Hence, $f \in \tilde{\boldsymbol{X}}$.

To compress this image, we notice that it consists of three smaller triangles which are reduced copies of itself. So, three contractive mappings are defined as

$$w_1\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

$$w_2\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \\ 0 \end{bmatrix}.$$

$$w_3\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \frac{1}{4} \\ \frac{1}{2} \end{bmatrix}$$

The w_1 is a simple shrinkage toward the origin. The w_2 is a shrinkage followed by a right-shift. The w_3 is a shrinkage followed by both right- and up-shift. These three mappings are depicted in Figure 3.4.

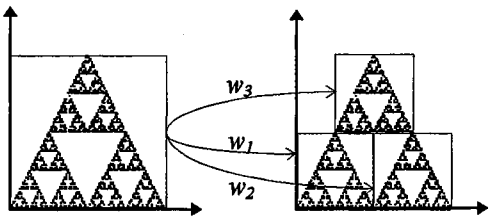


Figure 3.4 Three contractive mappings on the Sierpinski triangle.

It is obvious that these mappings are contractive, because the distance between any two points will be halved after one transformation w_i. Hence, the IFS, composed of w_1, w_2 and w_3,

$$\{w_i\}_{i=1}^{3}$$

$$W(B)=\bigcup_{i=1}^{3} w_i(B)$$

will have a fixed point image (attractor)

$$W(f)=f.$$

The attractor image is the Sierpinski triangle itself. So, three mappings can be recorded to represent this image.

The decompression is the course from any initial image to the attractor image. Figure 3.5 shows the first three decompression iterations. The original Sierpinski triangle will be reached eventually,

$$\lim_{n\to\infty} W^{(n)}(B)=f.$$

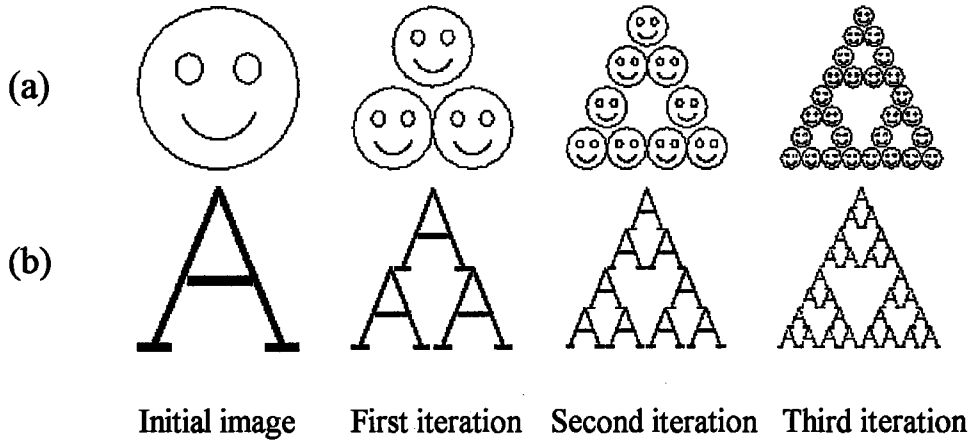


Figure 3.5 The decompression of Sierpinski triangle. The initial image can be anything such as (a) a smiling face, or (b) a letter A [Fisher95].

CHAPTER VI

PARTITIONED ITERATION FUNCTION SYSTEM AND CENTERING

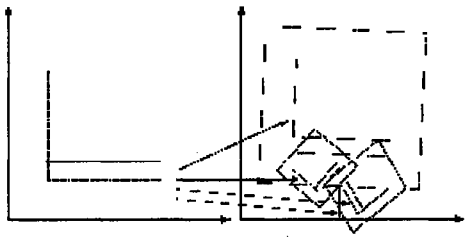
4.1 Introduction

In this chapter, we describe the partitioned iteration function system (PIFS) which is a generalization of the IFS to encode and decode grayscale images in section 4.2. PIFS was proposed by A. E. Jacquin in 1989 [Jacq89]. Similar to the IFS, the fixed point theorem of PIFS is shown in section 4.3. The coefficients of the contractive mappings are derived in section 4.4. Section 4.5 gives a PIFS compression example. We describe our centering model and discuss its advantage on decompression convergence in section 4.6. Section 4.7 shows the preliminary results and comparison with the PIFS model.

4.2 PIFS

Barnsley suggested that storing images as collections of transformations could lead to image compression. For example, the fern in Figure 4.1 is generated from only 4 contractive transformations.

The transformations looks like the ones in the Sierpinski triangle example in section 3.8, hence each transformation is defined by at most 6 numbers. Those 6 coefficients of each transformation do not require much memory to store on a computer. They can be stored in 4 transformation $\times$ 6 numbers/transformation $\times$ 32 bits/number = 768 bits = 96 bytes. The transformations can produce a fern image in arbitrary high resolution. However, it takes 1,048,576 bytes to store a 1024$\times$1024 image of the



(a) Four transformations of Barsley's fern

(b) the result of this IFS

Figure 4.1 (a) Four transformations of Barnsley's fern and (b) the result of this IFS.

enlarged fern as a collection of pixels. Barsley got the mythic compression ratio of 1,048,576/96 = 10,922 by this way.

Unfortunately, natural images are not exactly self-similar, i.e., it is unlikely to find a part of the image that is similar to the whole image. Based on self-similarity, Barnsley's IFS theory can do very little on natural image compression. But, in fact, an image does contain a different sort of self-similarity. Figure 4.2 shows sample regions of Clown that are similar at different scales: the two bulbs are almost identical, and the larger wall block after 50% reduction is similar to the smaller wall block.

With this observation, Jacquin modified Barnsley's IFS into a partitioned IFS which cuts the image into many smaller blocks and finds the similar pairs within blocks. Now, let us review the IFS defined in chapter III:

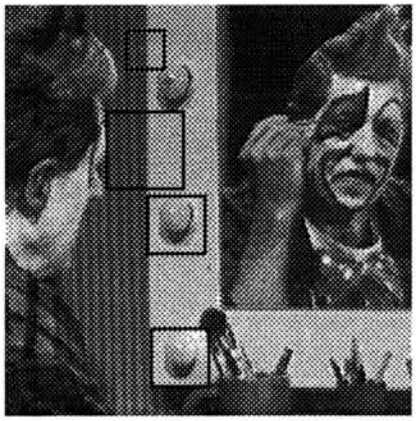


Figure 4.2 Self-similar portions of the Clown image.

$$W(A)=\bigcup_{i=1}^{N} w_i(A), \tag{4.1}$$

and the contractive transformation w_i can be defined as

$$w_i\begin{bmatrix} x \\ y \end{bmatrix}=\begin{bmatrix} M_i\cos\theta_i & -M_i\sin\theta_i \\ M_i\sin\theta_i & M_i\cos\theta_i \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}+\begin{bmatrix} S_{xi} \\ S_{yi} \end{bmatrix}. \tag{4.2}$$

We limit the contractive transformation to only evenly shrinking, rotation, and translation for our discussion, although skewing and stretching are also possible. The contractive transformation of the IFS contains the following components:

- the number of the original image pasted together to form the output, i.e., N;
- a setting of position, scaling, and rotation factors for each copy, i.e., S_{xi}, S_{yi}, M_i, and θ_i.

One of the deficiencies of the IFS is that there is no opportunity for a pixel to change its graylevel. Besides, the contractive transformation will only transform the

whole image into smaller blocks. To freedom these IFS's restrictions, the following two capabilities are added to IFS to derive PIFS:

- A contrast and brightness adjustment for each transformation,
- A mask which selects, for each transformation, a part of the original image to be transformed.

These extra features are sufficient to allow the encoding of grayscale images. The second feature is a new important feature. It partitions an image into pieces which are transformed individually. By partitioning the image into pieces, we allow the encoding of many shapes that are difficult to encode using an IFS. The trade-off of the partitioning is that a worse compression ratio will be obtained. The 10,000 to 1 myth is gone.

Let us review what happens when we encode an image using PIFS. Each mask selects a portion of the original image, whose support is denoted by D_i, and transforms that part of image above D_i with a contrast and brightness adjustment to another part of the image, whose support is denoted by R_i. We call the D_i domains and the R_i ranges. The transformation is denoted by w_i. When encoding, we find a piece of D_i and a mapping of w_i for each R_i, so that when we apply w_i to the part of the image over D_i, we get something that is very close to the part of the image over R_i. Finding the piece R_i with a surface which is as close as possible to the transformed surface of the corresponding D_i is the heart of fractal image compression.

The definition of PIFS is the same as IFS except that to the transformations w_i is added another dimension for changing the graylevel of the image. We denote the augmented w_i as v_i.

$$v_i\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}M_i\cos\theta_i & -M_i\sin\theta_i & 0\\ M_i\sin\theta_i & M_i\cos\theta_i & 0\\ 0 & 0 & \alpha_i\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}+\begin{bmatrix}S_{xi}\\S_{yi}\\\beta_i\end{bmatrix}, \tag{4.3}$$

where $z = f(x, y)$ is the graylevel of the pixel at the position (x, y), α_i controls the contrast, and β_i controls the brightness of the transformation. M_i, θ_i, S_{xi}, and S_{yi} determine how (by scaling, rotation, shifting, etc.) the partitioned domain D_i of an original image is mapped to the partitioned range R_i. We use a shorthand to denote the contractive transformation v_i of PIFS. Note that this contractive transformation contains two parts: graylevel transformation and support transformation:

$$R_i \leftarrow v_i(D_i). \tag{4.4}$$

Then the PIFS can be defined as

$$V(A)=\bigcup_{i=1}^{N} v_i(A). \tag{4.5}$$

Since we want $V(\cdot)$ to be an image, we must insist that $\bigcup_{i=1}^{N} R_i=A$ and that $R_i \bigcap R_j=\phi$ when $i \neq j$. That is, when we apply V to an image, we get some single valued function above each point of the square A.

4.3 Fixed points for PIFS

When decoding, we begin with an arbitrary initial image f_0 and then iterate

$$
\begin{aligned}
f_1 &= V(f_0) \\
f_2 &= V(f_1) = V(V(f_0)) = V^{(2)}(f_0) \\
&\cdots \\
f_n &= V^{(n)}(f_0)
\end{aligned}
\tag{4.6}
$$

Since V is contractive, eventually $f_\infty = V^{(\infty)}(f_0)$ will be a fixed-point image (attractor). In practical applications, the graylevels of an image are all quantized into integers. Hence, it will not take infinite steps to reach the attractor image.

4.4 Pairing R_i with the corresponding D_i

The transformation v_i contains two parts, the support transformation (w_i) and the graylevel transformation. The support transformation w_i determines how D_i is mapped to R_i. The graylevel transformation approximately transforms the graylevel z of a pixel above D_i at the location (x, y) into a new value close to the graylevel of a corresponding pixel above R_i. It is denoted as

$$
f(w(x,y)) \leftarrow \alpha_i \cdot f(x,y) + \beta_i, \qquad \forall_{(x,y)\in D_i}. \tag{4.7}
$$

Here, $f(x, y)$ represents the image surface above the D_i support, and

$w(x, y)$ represents the R_i support, i.e., the transformed D_i support, and

$f(w(x, y))$ represents the image surface above the R_i support.

We use a shorthand to rewrite it:

$$
r_i \leftarrow \alpha_i \cdot d_i + \beta_i, \tag{4.8}
$$

where $d_i \equiv f(x,y)$, $\forall_{(x,y)\in D_i}$, and

$r_i \equiv f(w(x,y))$.

The d_i values comprise the surface on D_i and represent the contents of the domain block. The r_i values comprise the surface on R_i and represent the contents of the range block. From now on, our discussion will be concentrated on this graylevel transformation.

The goal is to minimize the difference between the graylevels of range blocks and those of the transformed domain blocks. The least-squares error method is used to measure the difference. The graylevels of a transformed domain block are

$$\alpha_i \cdot d_i + \beta_i, \ \forall_{(x,y)\in D_i}. \tag{4.9}$$

The graylevels of the corresponding range block are

$$r_i, \qquad \forall_{(x,y)\in D_i}. \tag{4.10}$$

Thus, the square error (*SE*) between them is

$$SE = \iint_{(x,y)\in D_i} (\alpha_i \cdot d_i + \beta_i - r_i)^2 \, dxdy = \int (\alpha_i \cdot d_i + \beta_i - r_i)^2 \, dA. \tag{4.11}$$

The *SE* has its minimum value when the partial derivatives with respect to both coefficients (α and β) are all zero.

$$\begin{aligned} \frac{\partial}{\partial \alpha} \int (\alpha_i \cdot d_i + \beta_i - r_i)^2 dA &= 2 \cdot \int (\alpha_i \cdot d_i^2 + \beta_i \cdot d_i - r_i \cdot d_i) dA = 0 \\ \frac{\partial}{\partial \beta} \int (\alpha_i \cdot d_i + \beta_i - r_i)^2 dA &= 2 \cdot \int (\alpha_i \cdot d_i + \beta_i - r_i) dA = 0 \end{aligned} \tag{4.12}$$

Rewrite the above equations in matrix form:

$$\begin{bmatrix} \int d_i^2 dA & \int d_i dA \\ \int d_i dA & \int dA \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \int r_i \cdot d_i dA \\ \int r_i dA \end{bmatrix}. \tag{4.13}$$

This system of equations is solved to obtain α_i and β_i:

$$\alpha_i = \frac{\int dA \cdot \int d_i r_i dA - \int d_i dA \cdot \int r_i dA}{\int dA \cdot \int d_i^2 dA - \left(\int d_i dA\right)^2}$$
$$\beta_i = \frac{\int r_i dA - \alpha_i \cdot \int d_i dA}{\int dA} \qquad (4.14)$$

Then the minimized squared error (SE_i) is

$$SE_i = \int r_i^2 dA + \alpha_i\left(\alpha_i \int d_i^2 dA - 2\int d_i r_i dA + 2\beta_i \int d_i dA\right) + \beta_i\left(\beta_i \int dA - 2\int r_i dA\right) \qquad (4.15)$$

Since the size of the domain block (and/or range block) might be different, the better measurement of the difference between the range block and transformed domain block is the Root Mean Square (RMS_i) error,

$$RMS_i = \sqrt{\frac{SE_i}{\int dA}}. \qquad (4.16)$$

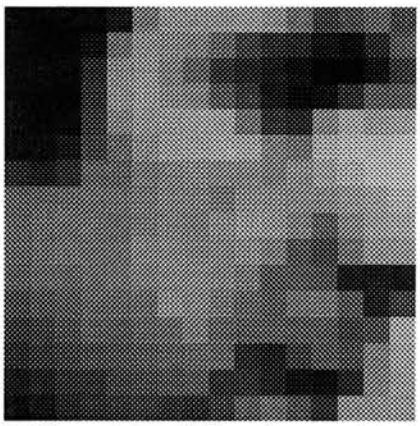
If the RMS_i error is small enough, i.e., less than a given value, the transformed d_i can pretty well represent r_i and the v_i transformation is recorded to represent r_i. If the RMS_i error is not good enough, we need to try other d_i until a good transformation for this r_i is found or no more d_i are available. If no d_i can be fairly transformed to match this r_i, the r_i needs to be broken down into smaller pieces and we repeat the matching process all over again for each smaller R_i. Fisher implemented a compression algorithm which calculates the α, β, and RMS with the previous method and uses quadtrees to break down the range block [Fish95]. The compression and decompression programs are listed and explained in [Fish95]. Readers also can use mosaic or lynx to get them: *http://inls3.ucsd.edu/Research/Fisher/Fractals/book.html*, under the link *enc.c*. All of the

experiments in this study are developed from and compared with these benchmark programs.

4.5 A PIFS example

Fisher implemented the PIFS model in his book [Fish95]. The program is called *fractal image compression with quadtrees* ,which will be discussed in detail in the next chapter. We use this program to compress a small piece of an image with dimension 16×16 and 256 graylevels per pixel. The demonstrated image is the photographer's face cropped from one of our test images, Camera. Figure 4.3 (a) is the original Camera image with a little white box shows the place to crop. Figure 4.3 (b) is the enlarged face image whose width and height are each 16 pixels.

(a) The 256×256 Camera image



(b) The enlarged face image, 16×16

Figure 4.3 (a) The 256x256 Camera image and (b) the 16×16 cropped and enlarged face image.

We set the minimum range block size to 4x4 and all domain block sizes to twice the range block size, 8×8. The program first partitions the input image into 16 range blocks, r_i, $i = 1..16$. (Figure 4.4).

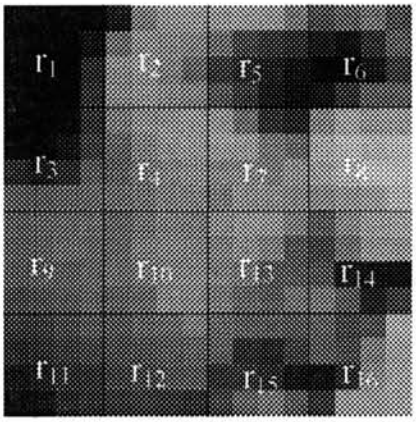


Figure 4.4 The range blocks

Then it partitions the input image into 4 domain blocks, d_i, $i = 1..4$ (Figure 4.5).

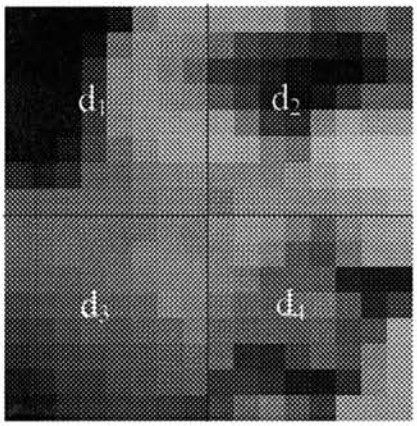


Figure 4.5 The domain blocks

Each range block r_i will be matched with a domain block d_i with a minimum RMS error. The searching process could be very time consuming if there are many range blocks and domain blocks. The results of the pairing are shown in Table 4.1.

TABLE 4.1

THE RANGE-DOMAIN PAIRING RESULTS

Range block	Domain block	Sym. operation	Quantized α	Quantized β
r_1	d_3	3	-0.5625	94.12
r_2	d_4	0	-0.375	218.11
r_3	d_3	7	-1	204.80
r_4	d_3	6	0.3125	131.14
r_5	d_3	5	-1	236.93
r_6	d_1	5	0.5	32.13
r_7	d_2	3	-0.375	215.34
r_8	d_4	3	-0.25	228.40
r_9	d_3	6	0.3125	99.52
r_{10}	d_2	1	-0.125	180.71
r_{11}	d_3	0	0.875	-31.12
r_{12}	d_3	0	0.5625	54.21
r_{13}	d_3	5	0.375	207.06
r_{14}	d_2	5	0.675	6.53
r_{15}	d_2	6	0.25	76.80
r_{16}	d_1	0	0.5	89.35

The Symmetry Operation in Table 4.1 is the set of 8 ways (4 rotations and a flip-flop) to align the range block and domain block. For example, symmetry operation equals 0 means

no rotation or flip-flop at all. Symmetry operation equals 1 means the upper-right corner of the range block matches the upper-left corner of the domain block, and the upper-left corner of the range block matches the bottom-left corner of the domain block.

The original file size is 256 bytes. The contractive mappings in Table 4.1 are stored as the representation of the compressed image. The compressed file size is 40 bytes. The decompressed image is enlarged and shown in Figure 4.6.

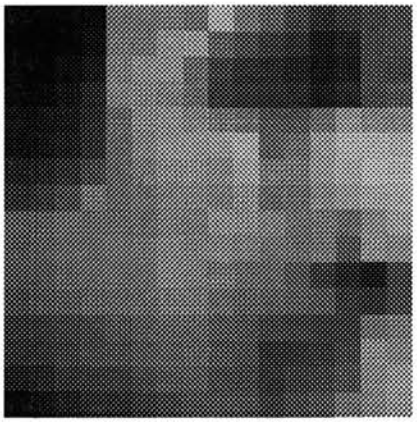


Figure 4.6 The decompressed face image (enlarged).

Figure 4.7 shows another approach which sets the minimum range block size to 2×2. There are 61 contractive transformations. The compressed file size, 152 bytes, is worse than the previous demonstration, 40 bytes, with the profit of better restored image fidelity.

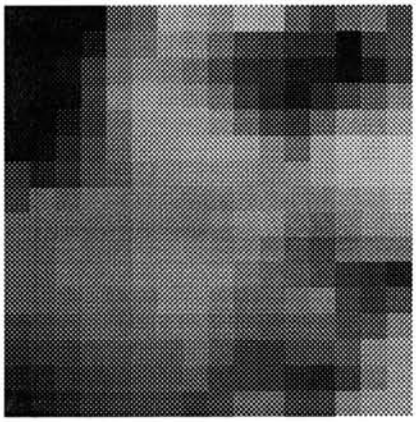


Figure 4.7 Another approach. Poorer compression, but better image quality.

4.6 Centering to Decorrelate α and β

Notice that the solutions of α and β are correlated. A larger α will result in a smaller β or even negative β. The correlation affects the distribution of α and β, hence their quantization is more complicated. In this section, we propose a model to decorrelate α and β by centering the r_i and d_i.

4.6.1 Compression with Centered PIFS

Equation 4.8 shows the contractive mapping of Jacquin's PIFS. The contractive mapping multiplies the image above the domain block by a contrast adjustment factor α, and then adapts it by a brightness adjustment factor β. The result should approximate the image above the range block.

In order to decorrelate α and β, the image above the domain block and the image above the range block should be centered. To center a block of the image, we first calculate the mean value of all pixels in that block. Then, from each pixel we subtract that

mean value. The contractive transformation that tries to pair a centered domain and centered range will be simplified so that only the multiplication of the contrast adjustment factor α is necessary. That is, the brightness adjustment factor β will have no effect. We show that β equals zero as follows. Let us modify the contractive mapping PIFS (Equation 4.8) into that of a centered PIFS:

$$(r_i - \bar{r}_i) \leftarrow \alpha_i \cdot (d_i - \bar{d}_i) + \beta_i, \tag{4.17}$$

where $\bar{r}_i$ is a flat image block with graylevel equal to the mean of the range block r_i,

$\bar{d}_i$ is a flat image block with graylevel equal to the mean of the domain block d_i.

Then α and β can be solved the same way as for Equation 4.8, except the d_i is now $(d_i - \bar{d}_i)$ and r_i is now $(r_i - \bar{r}_i)$. Equation 4.14 is the solution of Equation 4.8. We replace the d_i and r_i in Equation 4.14 with $(d_i - \bar{d}_i)$ and $(r_i - \bar{r}_i)$ respectively to derive our α_i solution. Let $\int dA = n$.

$$\alpha_i = \frac{\int dA \cdot \int (d_i - \bar{d}_i)(r_i - \bar{r}_i)dA - \int (d_i - \bar{d}_i)dA \cdot \int (r_i - \bar{r}_i)dA}{\int dA \cdot \int (d_i - \bar{d}_i)^2 dA - \left[\int (d_i - \bar{d}_i)dA\right]^2}$$

$$= \frac{n \cdot \int (d_i r_i - r_i \bar{d}_i - d_i \bar{r}_i + \overline{d_i r_i})dA - (\int d_i dA - n\bar{d}_i) \cdot (\int r_i dA - n\bar{r}_i)}{n \cdot \int (d_i^2 - 2\bar{d}_i d_i + \bar{d}_i^{\,2})dA - (\int d_i dA - n\bar{d}_i)^2}$$

$$
\begin{aligned}
&\frac{(n\cdot\int d_i r_i dA - n\overline{d_i}\cdot\int r_i dA - n\overline{r_i}\cdot\int d_i dA + n^2\overline{d_i r_i}) -}{(\int d_i dA\cdot\int r_i dA - n\overline{d_i}\cdot\int r_i dA - n\overline{r_i}\cdot\int d_i dA + n^2\overline{d_i r_i})} \\
&= \frac{}{n\cdot\int d_i^2 dA - 2n\overline{d_i}\cdot\int d_i dA + n^2\overline{d_i}^2 -} \\
&\qquad ((\int d_i dA)^2 - 2n\overline{d_i}\cdot\int d_i dA + n^2\overline{d_i}^2)
\end{aligned}
\tag{4.18}
$$

$$
= \frac{n\cdot\int d_i r_i dA - \int d_i dA\cdot\int r_i dA}{n\cdot\int d_i^2 dA - \left(\int d_i dA\right)^2}
$$

The α_i solution is the same as Jacquin's PIFS solution (Equation 4.14). That means the α_i calculated by Jacquin's PIFS is very meaningful. Its value best maps the domain block onto the range block. Other than that, the α_i calculation in Fisher's programs need not be changed.

To derive our β_i solution, we again replace the d_i and r_i in Equation 4.14 with $(d_i - \overline{d_i})$ and $(r_i - \overline{r_i})$ respectively:

$$
\begin{aligned}
\beta_i &= \frac{\int (r_i - \overline{r_i}) dA - \alpha_i \int (d_i - \overline{d_i}) dA}{n} \\
&= \frac{\int r_i dA - n\overline{r_i} - \alpha_i \left(\int d_i dA - n\overline{d_i}\right)}{n} \\
&= \frac{\int r_i dA}{n} - \overline{r_i} - \alpha_i \left(\frac{\int d_i dA}{n} - \overline{d_i}\right) \\
&= \overline{r_i} - \overline{r_i} - \alpha_i (\overline{d_i} - \overline{d_i}) \\
&= 0
\end{aligned}
\tag{4.19}
$$

The β_i value is always 0. In Fisher's programs, the β_i value is calculated by some real number multiplications and divisions. The real number β_i should then be quantized into an

integer. Our proposed centering model eliminates the β_i calculation and saves compression time.

Since $\beta_i = 0$, the contractive mapping of the centered PIFS can be rewritten as

$$(r_i - \bar{r}_i) \leftarrow \alpha_i \cdot (d_i - \bar{d}_i). \tag{4.20}$$

The goal of the contractive mapping is to approximate the range block r_i, so the right-hand side should be bare of r_i. We move the $\bar{r}_i$ to the right-hand side,

$$r_i \leftarrow \alpha_i \cdot (d_i - \bar{d}_i) + \bar{r}_i. \tag{4.21}$$

Now, the attractive transformation looks like a set of PIFS's with r_i as new β_i,

$$r_i \leftarrow \alpha_i \cdot (d_i - \bar{d}_i) + \beta_i', \text{ where } \beta_i' = \bar{r}_i. \tag{4.22}$$

Just as for the compression of PIFS model, the coefficients (α_i and β_i') of the transformation are recorded to represent the range block, r_i.

4.6.2 Decompression and Comparison

The decompression process for the centered PIFS is the same as that for the PIFS. The initial image can be anything, because the fixed point theorem guarantees an attractor image will be reached eventually. In one iteration of the PIFS model, every domain block d_i is multiplied by its corresponding α_i, and then added to its corresponding β_i. The resulting block is placed into the corresponding range block r_i location. In one iteration of the centered PIFS model, every domain block d_i is first centered by subtracting its mean value, then multiplied by its α_i and added to its β_i'. Just as for the decompression of PIFS model, the resulting block is placed into the corresponding range block r_i location.

Let us set the initial image all dark, i.e., the pixel values are all 0. For the PIFS model, the range block will be a flat β_i value at the end of the first iteration, because initially d_i is an all-zero flat block and the α_i multiplication contributes nothing. For the centered PIFS model, the range block will be a flat β_i' value at the end of the first iteration, because initially d_i and $\overline{d}_i$ are all-zero flat blocks and α_i multiplication contributes nothing. Note that $\beta_i' = \overline{r}_i$, that is, the range block becomes $\overline{r}_i$ (the range block mean value from the original image) at the end of the first iteration. With the collage of those mean-value range blocks, the decompression result of the first iteration should be recognizable.

Because β_i' is more meaningful than β_i, the centered PIFS creates a better image than the primitive PIFS does at the end of the first decompression iteration. That image is closer to the original image. We show how much closer it is compared with the result of the primitive PIFS.

First, we compute the difference ($diff_i$) between β_i and β_i':

$$\begin{aligned} diff_i &= \beta_i - \beta_i' \\ &= \beta_i - \overline{r}_i \end{aligned} \quad . \tag{4.23}$$

The solution of the β_i is carried out in Equation 4.14 and $\overline{r}_i$ is the mean value of the range block. So the $diff_i$ can be simplified as

$$\begin{aligned} diff_i &= \frac{\left(\int r_i dA - \alpha_i \int d_i dA\right)}{n} - \frac{\int r_i dA}{n} \\ &= \frac{\alpha_i \int d_i dA}{n} \\ &= \alpha_i \overline{d}_i \end{aligned} \quad . \tag{4.24}$$

The RMS_i error at the end of the first iteration in the centered PIFS model is

$$(RMS_i)_{centered-PIFS}=\sqrt{\frac{\int(r_i-\beta_i')^2 dA}{n}}. \tag{4.25}$$

This is actually the standard deviation of the range block:

$$(RMS_i)_{centered-PIFS}=\sqrt{\frac{\int(r_i-\bar{r}_i)^2 dA}{n}}. \tag{4.26}$$

The RMS_i error at the end of the first iteration in the primitive PIFS model is

$$\begin{aligned}
(RMS_i)_{PIFS} &= \sqrt{\frac{\int(r_i-\beta_i)^2 dA}{n}} \\
&= \sqrt{\frac{\int(r_i-(\bar{r}_i+diff_i))^2 dA}{n}} \\
&= \sqrt{\frac{\int((r_i-\bar{r}_i)-diff_i)^2 dA}{n}} \\
&= \sqrt{\frac{\int((r_i-\bar{r}_i)^2-2diff_i(r_i-\bar{r}_i)+diff_i^2)dA}{n}} \\
&= \sqrt{\frac{\int((r_i-\bar{r}_i)^2+diff_i^2)dA-2diff_i\int(r_i-\bar{r}_i)dA}{n}} \\
&= \sqrt{\frac{\int((r_i-\bar{r}_i)^2+diff_i^2)dA}{n}-2diff_i\left(\frac{\int r_i dA}{n}-\frac{\int \bar{r}_i dA}{n}\right)} \\
&= \sqrt{\frac{\int((r_i-\bar{r}_i)^2+diff_i^2)dA}{n}-2diff_i\left(\bar{r}_i-\frac{n\bar{r}_i}{n}\right)} \\
&= \sqrt{\frac{\int((r_i-\bar{r}_i)^2+(\alpha_i\bar{d}_i)^2)dA}{n}}
\end{aligned} \tag{4.27}$$

Compare Equation 4.26 with Equation 4.27. The right-hand side of Equation 4.26 is less than the right-hand side Equation 4.27:

$$\sqrt{\frac{\int (r_i - \bar{r}_i)^2 dA}{n}} \le \sqrt{\frac{\int \left((r_i - \bar{r}_i)^2 + (\alpha_i \bar{d}_i)^2 \right) dA}{n}}. \qquad (4.28)$$

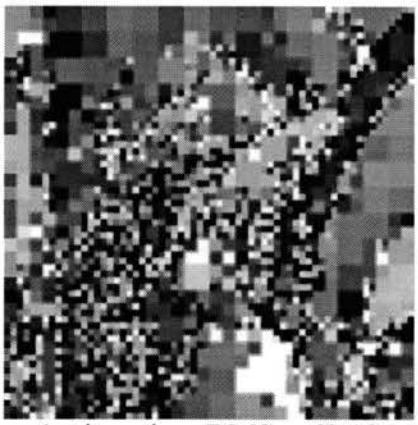
This proves the primitive PIFS's RMS error at the end of the first iteration is worse than for the centered PIFS's:

$$(RMS_i)_{centered-PIFS} \le (RMS_i)_{PIFS}. \qquad (4.29)$$

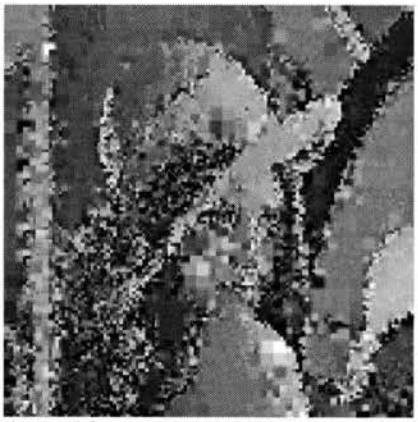
4.7 Experimental Results and Convergence Comparisons

The experiments are based on Fisher's *fractal image compression with quadtrees* programs which we will discuss in detail in the next chapter. The compression and decompression programs are in the appendix of [Fish95] and can be obtained by mosaic or lynx to *http://inls3.ucsd.edu/Research/Fisher/Fractals/book.html*, under the link *enc.c* and *dec.c*. The tested grayscale images are Lena, Clown, Camera, Hamlet and Kgirl which can be ftp'ed from *eedsp.gatech.edu:/database/images* directory. The image dimensions are 256×256 with 256 graylevels (8 bits) for each pixel, i.e., the file size of the original image is 256×256 = 65,536 bytes. The decompression sequences of Lena using the primitive PIFS and our centered PIFS methods are in Figure 4.8.

Fisher's PIFS



1st iteration, RMS = 62.19



2nd iteration, RMS = 35.95

3rd iteration, RMS = 21.89

Centered PIFS

1st iteration, RMS = 15.74

2nd iteration, RMS = 11.33

3rd iteration, RMS = 8.98

Figure 4.8 Decompression sequences of Lena using PIFS and centered PIFS.

4th iteration, RMS = 12.28

4th iteration, RMS = 8.88

5th iteration, RMS = 9.48

5th iteration, RMS = 8.87

6th iteration, RMS = 8.97

6th iteration, RMS = 8.87

Figure 4.8: Decompression sequences of Lena using PIFS and centered PIFS. (continued)

7th iteration, RMS = 8.90

7th iteration, RMS = 8.87

8th iteration, RMS = 8.89

8th iteration, RMS = 8.87

9th iteration, RMS = 8.88

9th iteration, RMS = 8.87

Figure 4.8: Decompression sequences of Lena using PIFS and centered PIFS. (continued)

The comparison curves of the decoding convergence for both methods are in Figure 4.9 for Lena. The comparison curves for Clown, Camera, Hamlet and Kgirl are in Figure 4.10, 4.11, 4.12 and 4.13 respectively. Their corresponding comparative decompression sequences are in Appendix A to E.

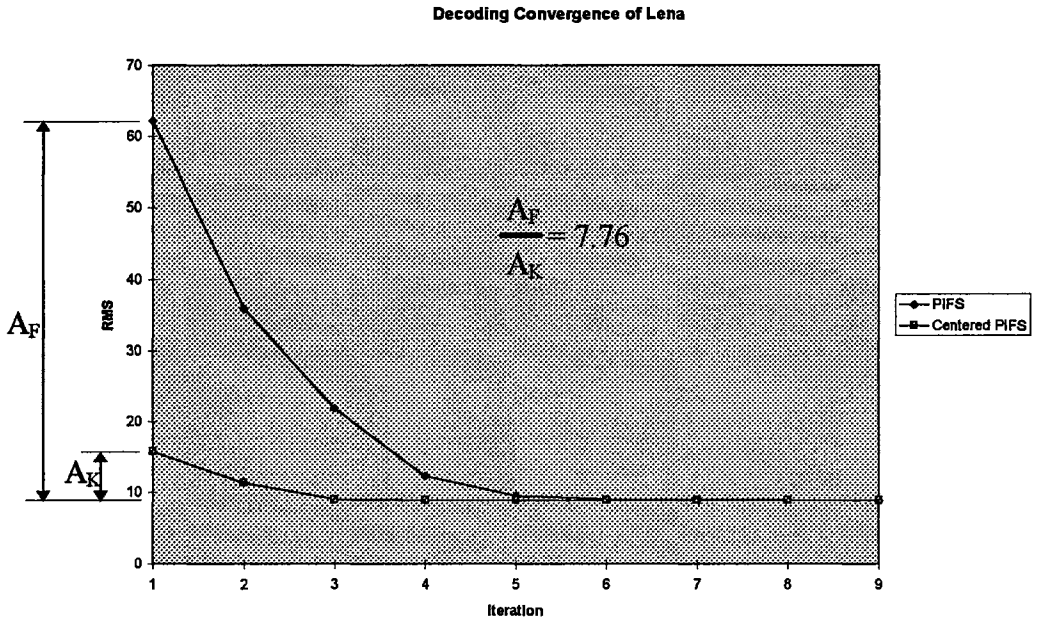


Figure 4.9 Decoding convergence of Lena using PIFS and centered PIFS

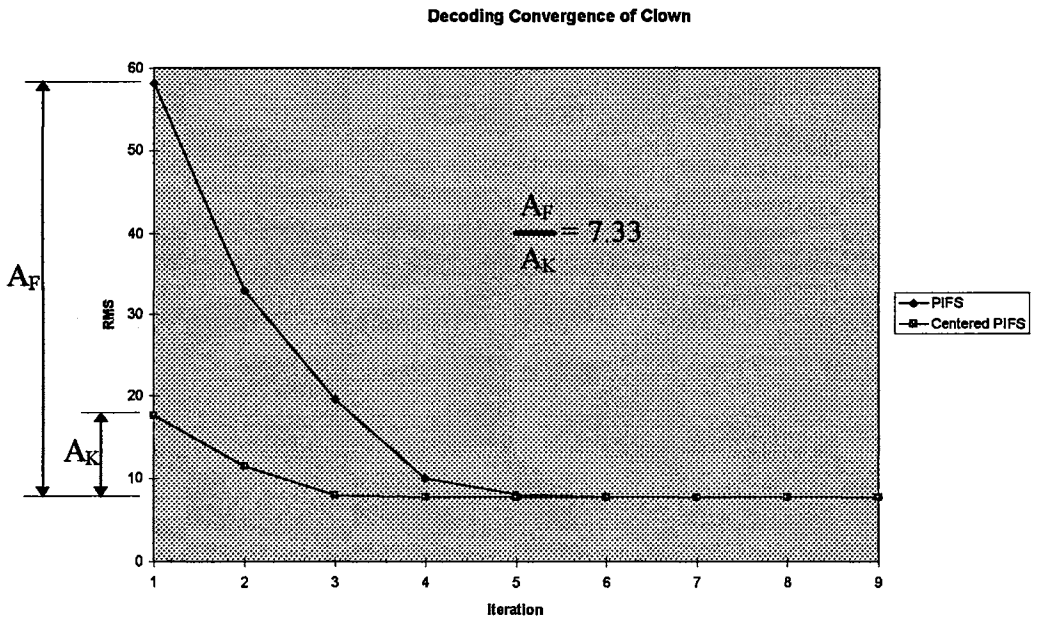


Figure 4.10 Decoding convergence of Clown using PIFS and centered PIFS

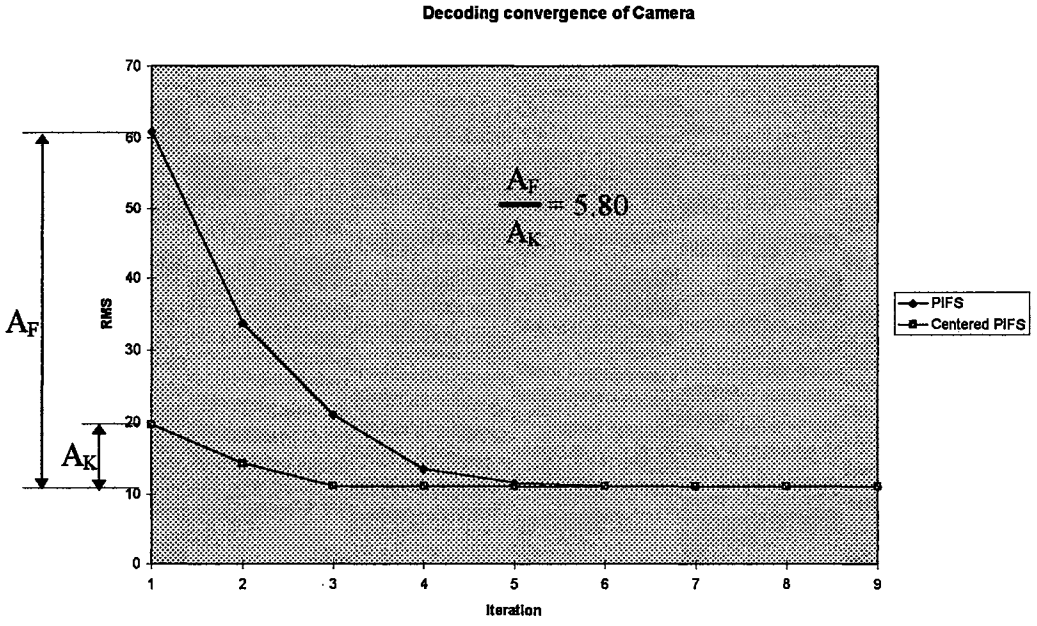


Figure 4.11 Decoding convergence of Camera using PIFS and centered PIFS

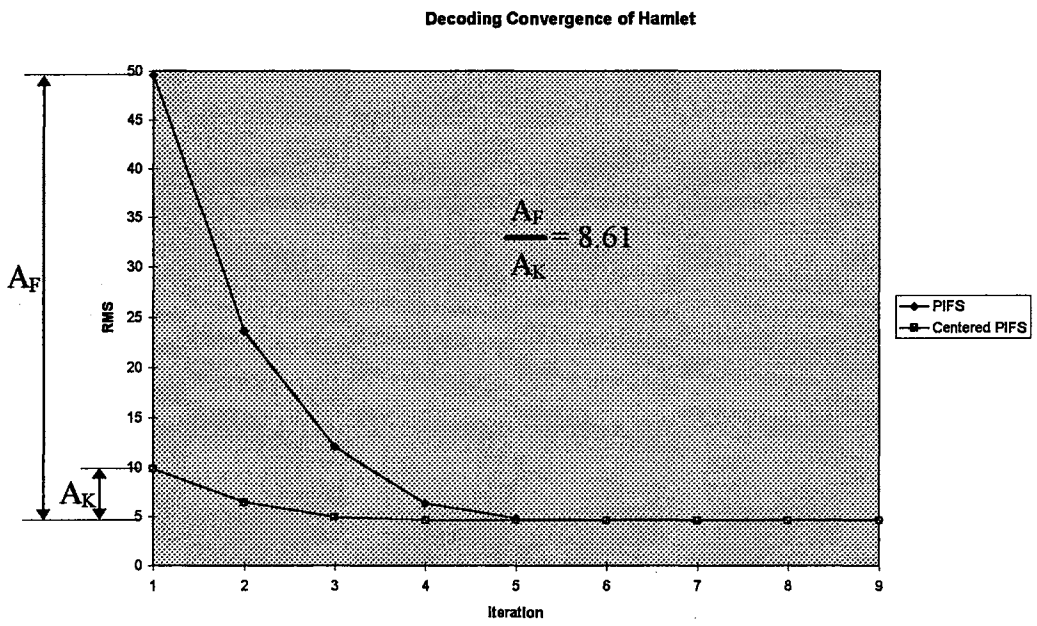


Figure 4.12 Decoding convergence of Hemlet using PIFS and centered PIFS

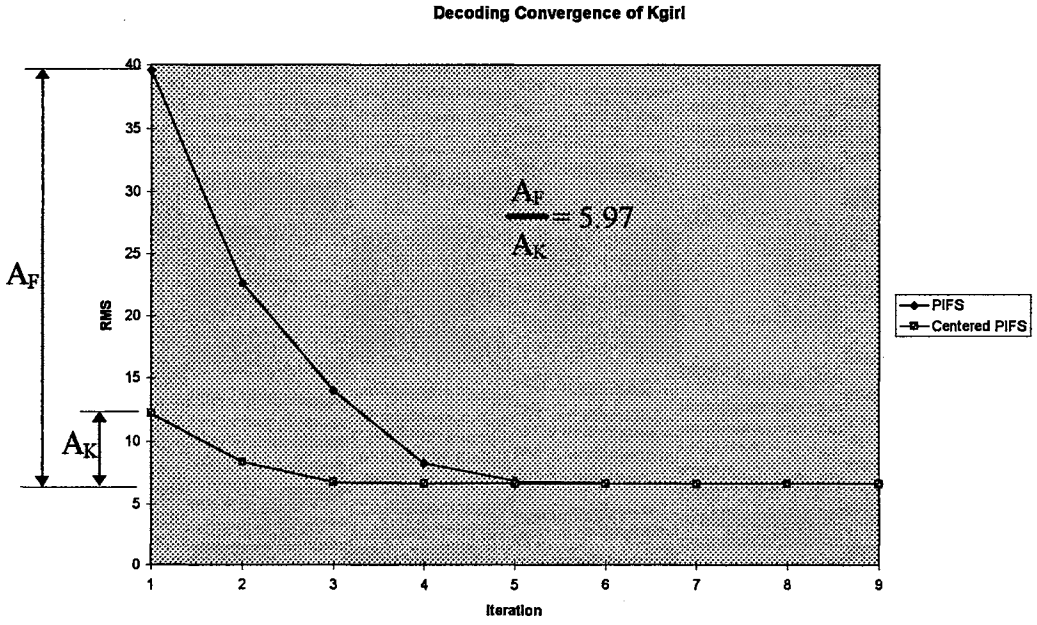


Figure 4.13 Decoding convergence of Kgirl using PIFS and centered PIFS

The domain and range blocks are centered before the pairing process, to decorrelate the coefficients of the contractive mapping. This decorrelation helps little on the compression ratio, because there are still two coefficients to be recorded to represent the graylevel transformation. However, the distributions of the decorrelated α_i and β_i' are more explicit so that more precise quantization is possible. Precise quantization could result in slightly better decompressed image fidelity. These aspects are discussed in Chapter VI.

The major advantage of the decorrelation is the convergence speed of the decompression sequences. See Figure 4.8. According to our decompression experiments, the primitive PIFS method usually takes 8 to 10 iterations to reach the attractor image, while the centered PIFS method can converge to the attractor image in 4 to 6 iterations. Besides, the images are always recognizable at the end of the first decompression iteration. It is due to this win at the first iteration that the centered PIFS enjoys faster convergence speed. Furthermore, the images at the end of the second iteration are almost indistinguishable from the final attractor image.

CHAPTER V

FRACTAL IMAGE COMPRESSION WITH QUADTREES, BATH FRACTAL TRANSFORMATION AND HYBRID ALGORITHM

Fisher extended Jacquin's [Jacq89] work and developed a quadtree-based fractal encoding scheme. It is a typical fractal scheme and a good first step for those who wish to implement their own. Hence, this scheme becomes a good reference point for other fractal schemes. Our study is based on this scheme too. We also compare our results with Fisher's results. The compression and decompression programs of this scheme are given in Appendix A of [Fish95] and stored at *http://inls3.ucsd.edu/Research/Fisher/Fractals/book.html* under the links *enc.c* and *dec.c*.

5.1 Encoding

The pseudo-code of the encoding steps that targets an image fidelity e_c is as follows [Fish95]:

- Choose a tolerance level e_c.
- Set r_1 as the whole image and mark it uncovered.
- While there are uncovered ranges r_i do {
    - Out of the possible domains d, find the domain d_i and the corresponding v_i that best cover r_i (i.e., that minimizes the RMS error).
    - If (RMS < e_c) or (size(r_i) ≤ r_{min}) then

- Mark r_i as covered, and write out the transformation v_i;

- else

  - Partition r_i into smaller ranges that are marked as uncovered, and remove r_i from the list of uncovered ranges.

}

In this case, the collection of ranges comes from a quadtree partition of the image, and the collection of domains consists of subsquares of the image that are twice the range size.

5.1.1 The Ranges

A quadtree partition is a representation of an image as a tree in which each node, corresponding to a square portion of the image, contains zero or four subnodes, corresponding to the four quadrants of the square. The root of the tree is the initial image. A node containing zero subnodes is called a leaf node.

The ranges are selected as follows: after some initial number of quadtree partitions are made (corresponding to a minimum tree depth), the squares at the nodes are compared with domains from the domain library (or domain pool) d. Only the domains with a size of twice the range size are compared. The pixels in the domain are averaged in groups of four so that the domain is reduced to the size of the range, and the affine transformation (v_i) of the pixel values (i.e., a scaling (α_i) and offset (β_i)) is calculated that minimizes the RMS difference between the transformed domain pixel values and the range pixel values. All of the potential domains are compared with a range, one by one, until the RMS value is below a preselected threshold, e_c. Then the optimal domain and the affine

transformation are stored -- this constitutes one map v_i. The collection of all such maps $V = \bigcup v_i$ constitutes the encoding. If the resulting optimal RMS value is above the threshold and if the depth of the quadtree is less than a preselected maximum tree depth, then the range square is subdivided into four quadrants (which means adding four subnodes to that range node), and the process is repeated.

5.1.2 The Domains

The large number of domain blocks is the major cause of slowness, if each range block is compared with each domain block. Suppose the image is partitioned in to 4×4 range blocks. A 256×256 image contains a total of $(256-8+1)^2 = 62{,}001$ different 8×8 domain blocks. If the 8 isometric symmetries are included, the total number of domain blocks increases to 496,008. There are $(256-4+1)^2 = 64{,}009$ 4×4 range blocks, which makes for a maximum of 64,009×496,008 = 31,748,976,072 possible pairings to test! The exhaustive pairing cannot be completed in three days on a 486-66 MHz PC.

In our study, the domain pool is constructed with a restriction: non-overlapping blocks with dimension twice the range size. Hence, a 256×256 image contains only $(256/8)^2 = 1{,}024$ different 8×8 domain blocks.

It is important to note that each domain can be mapped onto a range in eight different ways -- four rotations and a flip with four more rotations. Thus, the domain pool can be thought of as containing the domains in eight possible orientations. In practice, the classification scheme discussed below defines a fixed rotation, so that the domains are only considered in one or two orientations.

5.1.3 The Classification

The domain-range comparison step of the encoding is very computationally intensive. A classification scheme is used in order to minimize the number of domains compared with a range. Before the encoding, all the domains in the domain library are classified; this avoids reclassification of domains. During the encoding, a potential range is classified, and only domains with the same (or near) classification are compared with the range. This significantly reduces the number of domain-range comparisons. By "near" classifications it means squares that would have been classified differently if their pixel values were slightly different. The idea of using a classification scheme was developed independently in [Jacq89] and [JaBoFi89].

Many classification schemes are possible. For example, Jacquin used a scheme that classified a sub-image into flat, edge, and texture regions. Here, a scheme similar to the one presented in [FiJaBo91] is discussed. A square sub-image is divided into upper left, upper right, lower left, and lower right quadrants, which are numbered sequentially. On each quadrant, we compute values proportional to the average and the variance. If the pixel values in quadrant i are $p^i_1, \ldots, p^i_n$ for $i = 1,2,3,4$, we compute

$$A_i = \sum_{j=1}^{n} p^i_j \tag{5.1}$$

and

$$VAR_i = \sum_{j=1}^{n} \left(p^i_j\right)^2 - A_i^2. \tag{5.2}$$

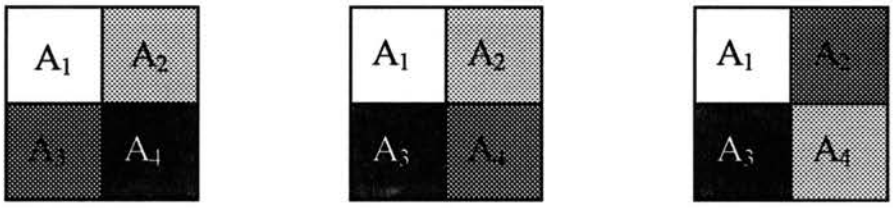
It is always possible to orient the sub-image so that the A_i's are ordered in one of the following three ways:

Major Class 1: $A_1 \geq A_2 \geq A_3 \geq A_4$.

Major Class 2: $A_1 \geq A_2 \geq A_4 \geq A_3$.

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

These classifications correspond to the brightness levels shown in Figure 5.1. Once the rotation of the square has been fixed, each of the 3 major classes has 24 subclasses consisting of the 24 orderings of the VAR_i. Thus, there are 72 classes in all. If the scaling value α_i is negative, the orderings in the classes above are rearranged. Therefore, each domain is classified in two orientations, one orientation for positive α_i and the other for negative α_i.



Major Class 1 Major Class 2 Major Class 3

Figure 5.1 Orientation of a square image. The brightness of quadrants can be arranged (rotation and flip-flop) into one of these three canonical classes.

Hence, an encoding of an image consists of the following data:

- The final quadtree partition of the image
- The scaling and offset values α_i and β_i for each range
- For each range, a domain that is mapped to it

- The symmetry operation (orientation) used to map the domain pixels onto the range pixels

5.1.4 The Quantization

The affine transformation on the pixel values consists of a multiplication by a scaling factor α_i and an addition of an offset value β_i (these can be thought of as a contrast and brightness adjustment). These values must be quantized for efficient storage, and it is these quantized values that are used when computing the RMS difference between the domain block and the range block.

If the magnitude of the optimal scaling factor $|\alpha_i|$ is greater than 1 for any of the v_i, there is a chance that the resulting map V will not be eventually contractive. Thus, $|\alpha_i|$ values larger than some maximal value α_{max} are truncated. In this study, α_{max} is set to 1.0.

5.2 Decoding

Decoding an image consists of iterating V from any initial image. The quadtree partition is used to determine all the ranges in the image. For each range R_i, the domain D_i that maps to it is shrunk by half in each dimension by averaging nonoverlapping groups of 2×2 pixels. The shrunken domain pixel values are then multiplied by α_i, added to β_i, and placed in the location in the range determined by the orientation information. Completion of all v_i contractive mappings constitutes one decoding iteration. The decoding step is iterated until the attractor image is approximated; i.e., until further iteration does not change the image or until the change is below some small threshold value.

5.3 Postprocessing

Since the ranges are encoded independently, there is no guarantee that the pixel values will be smooth at the block boundaries. Human eyes are sensitive to such discontinuities, even when they are small. It is possible to minimize these artifacts by postprocessing the image. In Fisher's decompression program, only the pixels at the boundary of the range blocks are modified, using a weighted average of their values. If the pixel values on either side of a boundary are p_a and p_b, then these are replaced by $w_1p_a+w_2p_b$ and $w_2p_a+w_1p_b$, with $w_1+w_2 = 1$. Ranges that occur at the maximum depth of the quadtree are averaged with weights of $w_1 = 5/6$ and $w_2 = 1/6$, and those above this depth are averaged with weights of $w_1 = 2/3$ and $w_2 = 1/3$. These values are largely heuristic (as is the method itself), but the results seem satisfactory.

5.4 Efficient Storage

To increase the compression, the following bit allocation schemes are used.

Quadtree: One bit is used at each quadtree level to denote a further recursion or subsequent transformation information. At the maximum depth, however, no such bit is used, since the decoder knows that no further division is possible.

Scaling (α_i) and Offset (β_i): Five bits are used to store the scaling and seven bits for the offset. No form of adaptive coding is done on these coefficients, even though their distributions show considerable structure.

Domains: The domains are indexed and referenced by this index. However, when the scaling value (α_i) is zero, the domain is irrelevant, and so no domain or orientation information is stored in this case.

Orientation: Three bits are used to store the orientation of the domain-range mapping.

5.5 Bath Fractal Transformation

The graylevel transformation of Jacquin's PIFS transforms the graylevel of a pixel at location *(x, y)* by multiplying it by a contrast factor α_i and shifting it by a brightness factor β_i. The resulting graylevel approximates the graylevel of the corresponding pixel of the range block.

$$f(w(x,y)) = \alpha_i f(x,y) + \beta_i, \qquad \forall (x,y) \in D_i. \tag{5.3}$$

We rewrite this in a shorthand

$$r_i \leftarrow \alpha_i \cdot d_i + \beta_i. \tag{5.4}$$

Monro called this an order-0 transformation, because only a constant term β is used, i.e., it does not utilize the x and y variables directly. In 1992, Monro and Dudbridge from the University of Bath, England, generalized Jacquin's scheme by adding an x term and a y term into the graylevel transformation and called it Bath Fractal Transformation (BFT):

$$r_i \leftarrow \alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y. \tag{5.5}$$

Similar to the PIFS model, the difference between the graylevels of a range block and those of the transformed domain block should be minimized. Monro also used the least-squares error method to minimize the difference. The graylevels of a transformed domain block are

$$\alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y, \qquad \forall_{(x,y)\in D_i}. \tag{5.6}$$

The graylevels of the correspondent range block are

$$r_i, \qquad \forall_{(x,y)\in D_i}. \tag{5.7}$$

Thus, the squared error (SE_i) between them is

$$\begin{aligned} SE_i &= \iint_{(x,y)\in D_i} \left(\alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y - r_i\right)^2 dxdy \\ &= \int \left(\alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y - r_i\right)^2 dA \end{aligned} \tag{5.8}$$

The SE_i has the minimum value when the partial derivatives with respect to each coefficient (α_i, β_i, a_1 and b_1) are all zero,

$$\begin{aligned} &\frac{\partial}{\partial \alpha_i} \int \left(\alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y - r_i\right)^2 dA = 0 \\ &\frac{\partial}{\partial \beta_i} \int \left(\alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y - r_i\right)^2 dA = 0 \\ &\frac{\partial}{\partial a_1} \int \left(\alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y - r_i\right)^2 dA = 0 \\ &\frac{\partial}{\partial b_1} \int \left(\alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y - r_i\right)^2 dA = 0 \end{aligned} \tag{5.9}$$

Hence

$$\begin{aligned} &2 \cdot \int \left(\alpha_i \cdot d_i^2 + \beta_i d_i + a_1 x d_i + b_1 y d_i - r_i d_i\right) dA = 0 \\ &2 \cdot \int \left(\alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y - r_i\right) dA = 0 \\ &2 \cdot \int \left(\alpha_i \cdot d_i x + \beta_i x + a_1 x^2 + b_1 xy - x r_i\right) dA = 0 \\ &2 \cdot \int \left(\alpha_i \cdot d_i y + \beta_i y + a_1 xy + b_1 y^2 - y r_i\right) dA = 0 \end{aligned} \tag{5.10}$$

Rewrite the above equations to the following matrix form:

$$\begin{bmatrix} \int x^2 & \int xy & \int xg(x,y) & \int x \\ \int xy & \int y^2 & \int yg(x,y) & \int y \\ \int xg(x,y) & \int yg(x,y) & \int g(x,y)^2 & \int g(x,y) \\ \int x & \int y & \int g(x,y) & \int 1 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ s \\ r \end{bmatrix} = \begin{bmatrix} \int xg(v(x,y)) \\ \int yg(v(x,y)) \\ \int g(x,y)g(v(x,y)) \\ \int g(v(x,y)) \end{bmatrix} \tag{5.11}$$

This system of equations is solved to obtain α_i, β_i, a_1 and b_1. Those transformation coefficients along with the location of the domain block can be stored to represent the range block.

The result of Monro's order-1 BFT showed an inferior compression-fidelity tradeoff compared to the ADCT (adaptive discrete cosine transform) method. However, Monro expected further work could yield better results [MoDu92b].

In 1993, based on the order-1 BFT, Monro implemented order-2 and order-3 BFTs with different levels of local search [Monr93].

Order-2 BFT:

$$r_i \leftarrow \alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y + a_2 x^2 + b_2 y^2 \tag{5.12}$$

Order-3 BFT:

$$r_i \leftarrow \alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + a_3 x^3 + b_3 y^3 \tag{5.13}$$

The method to solve those coefficients is similar to that of order-1 BFT.

When range blocks are searching for a similar corresponding domain block, Monro, based on his intuition, assumes that the similar blocks usually flock together (local

self-similarity). Hence, a local search should be enough to find a best match. The results show that the fidelity of the fractal transform increases with both search level and order of the polynomial approximation. Monro concluded that the order-1 BFT without searching gives a lower RMS error than Jacquin's order-0 model does with level 6 searching. Level 6 searching means domain blocks are at most 6 pixels away from the range block. The order-2 case also provides a notable improvement, beyond which there appears to be little improvement in decompressed image fidelity. Later, the local self-similarity was proven wrong by Fisher [Fish95], i.e., the successfully matched domains are essentially random and there is no preference for local domains.

In 1994, based on [Monr93], Monro examined BFTs of order-0, order-1 and order-2 without searching and evaluated the degradation introduced by quantization of the coefficients of contractive mappings [MoWo94]. Higher orders and local searching were not considered in this investigation because earlier results in [Monr93] indicated that little was gained by an order-3 approximation or local searching. In Figure 5.2, the result shows at the high fidelity/low compression end of the curve, higher order methods are better in the sense that lower error rates are observed at a given compression ratio. Surprisingly, at higher compressions, the lower order BFT performs better.

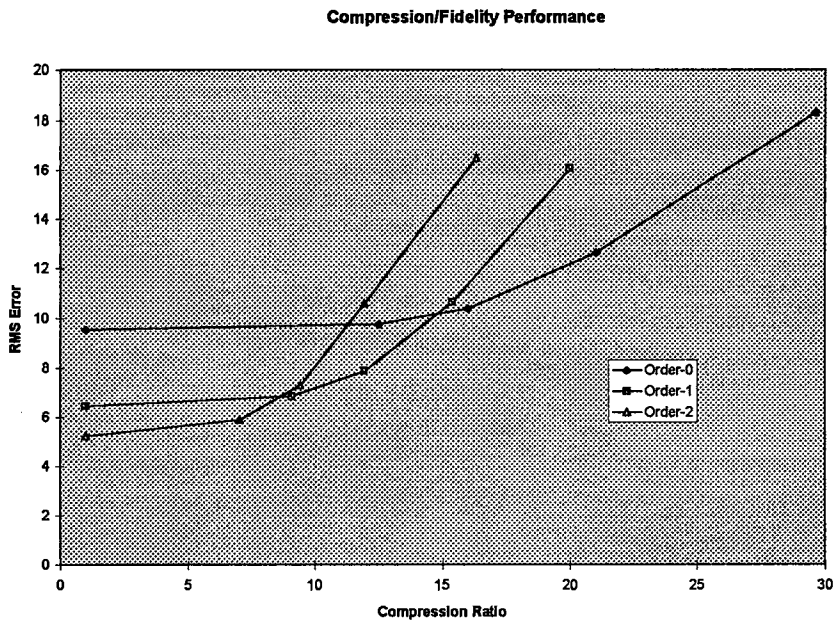


Figure 5.2 Compression/fidelity performance of three orders of the BFT without searching [MoWo94]

5.6 Centered BFT with quadtrees

The compression time for traditional fractal image compression is longer than for decompression. The advantage of Monro's higher order BFT is that the compression time and decompression time are nearly symmetrical, because no domain search is necessary during compression. However, the disadvantage of higher order BFTs is the inferior compression-fidelity tradeoff.

Jacquin's PIFS is an order-0 case of the BFT with searching. The advantage of PIFS is the better compression-fidelity tradeoff. Contrarily, the disadvantage of PIFS is the slow compression, because further partitioning on a range block is needed when the searching fails to find a good enough domain block for that range block. Fisher implemented a quadtree partitioning method in [Fish95]. The quadtree partitioning

produces four more range blocks to be compressed and each new range block will cause the searching to start all over again.

We propose a hybrid model that is a combination of Fisher's and Monro's. Our model eliminates the disadvantages from both models and enjoys the advantages from both models.

5.6.1 BFT with *xy* cross terms

The graylevel transformation of Monro's BFT does not include the *xy* cross terms which provide a rotation transformation of the image surface constructed by the polynomial terms. In order to make our model a complete one, we suggest including the *xy* cross terms. The following is the graylevel transformation with *xy* cross terms.

$$r_i \leftarrow \alpha_i \cdot d_i + \beta_i + a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + c_{11} xy + a_3 x^3 + b_3 y^3 + c_{21} x^2 y + c_{12} x y^2 + \cdots \tag{5.14}$$

Compared with order-0, the order-1 has two more coefficients, a_1 and b_1. Compared with order-1, the order-2 has three more coefficients, a_2, b_2 and c_{11}. Compared with order-2, the order-3 has four more coefficients, a_3, b_3, c_{12} and c_{21}.

5.6.2 Centering the image on domain and range blocks to decorrelate α_i and the polynomial

Similarly, the solutions of α_i and the polynomial are correlated. We decorrelate α_i and the polynomial by centering the r_i and d_i.

$$(r_i - \bar{r}_i) \leftarrow \alpha_i (d_i - \bar{d}_i) + \beta_i + a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + c_{11} xy + \cdots \tag{5.15}$$

The coefficients, α_i, β_i, a_1, b_1, a_2, b_2, c_{11}, etc., can be derived using the same method as in section 4.4.

5.6.3 Re-centering the polynomial

Since, the summation of the centered image block should be always zero, i.e., $\int (r_i - \overline{r_i}) dA = 0$ and $\int (d_i - \overline{d_i}) dA = 0$, the summation of the polynomial (the rest of the contractive mapping) should also be zero:

$$\int \left(\beta_i + a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + c_{11} xy + \cdots \right) dA = 0 \tag{5.16}$$

However, after β_i, a_1, b_1, a_2, b_2 and c_{11} are quantized, $\int \left(\beta_i + a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + c_{11} xy + \cdots \right) dA$ may no longer be zero. So, the proper β_i value should be adjusted to guarantee that the summation of the polynomial is zero. Let us denote the new β_i as $\tilde{\beta}_i$:

$$\begin{aligned}
&\int \left(\tilde{\beta}_i + a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + c_{11} xy + \cdots \right) dA = 0 \\
&\Rightarrow \int \tilde{\beta}_i dA + \int \left(a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + c_{11} xy + \cdots \right) dA = 0 \\
&\Rightarrow n \cdot \tilde{\beta}_i = -\int \left(a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + c_{11} xy + \cdots \right) dA \\
&\Rightarrow \tilde{\beta}_i = \frac{-\int \left(a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + c_{11} xy + \cdots \right) dA}{n}
\end{aligned} \tag{5.17}$$

The contractive mapping of the centered BFT with xy cross-terms can be rewritten as

$$(r_i-\bar{r}_i)\leftarrow-\alpha_i(d_i-\bar{d}_i)+\tilde{\beta}_i+a_1x+b_1y+a_2x^2+b_2y^2+c_{11}xy+\cdots. \quad (5.18)$$

The goal of the contractive mapping is to approximate the range block r_i, so that the right-hand side should be just r_i. We move the $\bar{r}_i$ to the left-hand side and let β_i' equal the summation of r_i and $\tilde{\beta}_i$:

$$r_i \leftarrow \alpha_i(d_i-\bar{d}_i)+\beta_i'+a_1x+b_1y+a_2x^2+b_2y^2+c_{11}xy+\cdots, \quad (5.19)$$

where $\beta_i'=\tilde{\beta}_i+\bar{r}_i$.

5.6.4 Algorithm of centered BFT with quadtrees

We propose an algorithm which targets an image fidelity e_c as follows. The algorithm adaptively uses a higher order BFT. The centered range block tries the pairing with centered domain blocks using order-0 BFT first. If the best-match domain block is still worse than our target fidelity e_c, higher order BFTs are applied to this best match domain block until the image fidelity is satisfied or the maximum order of the BFT is reached. If even the maximum-order BFT still cannot make a satisfactory pairing, the range block is inevitably partitioned into several smaller blocks which should make it easier to find a centered domain block that will achieve the target fidelity. The pseudo-code of the encoding steps is as follows:

- Choose a tolerance level e_c.
- Choose a maximum order n_{max}.
- Set r_1 as the whole image and mark it uncovered.

- While there are uncovered ranges r_i do {
    - Out of the possible domains d, find the best domain d_i and the corresponding v_i of order-0 BFT that best cover r_i (i.e., that minimizes the RMS error).
    - If (RMS $< e_c$) or (size(r_i) $\leq r_{min}$) then
        - Mark r_i as covered, and write out the transformation v_i;
    - else
        - set $n = 0$
        - {
            - $n = n+1$
            - Use that d_i to calculate the corresponding v_i of an order-n BFT that best covers r_i (i.e., that minimizes the RMS error)
        - } Repeat until ($n \geq n_{max}$) or (RMS $< e_c$)
        - If (RMS $< e_c$) then
            - Mark r_i as covered, and write out the transformation v_i of order-n BFT;
        - else
            - Partition r_i into smaller ranges that are marked as uncovered, and remove r_i from the list of uncovered ranges.

}

In our implementation, the collection of range blocks comes from a quadtree partitioning of the image, and the collection of domain blocks consists of subsquares of the

image that are twice the range block size. The experimental results are listed and discussed in chapter VI.

The decoding algorithm is basically identical to that of original PIFS model, except that the domain blocks should be centered before being multiplied by α_i.

CHAPTER VI

EXPERIMENTAL RESULTS AND COMPARISONS

The source programs of the compression and decompression are based on Fisher's *fractal image compression with quadtrees*. The source programs are listed in the Appendix A of [Fish95] and in the URL: *http://inls3.ucsd.edu/Research/Fisher/Fractals/book.html* under the links *enc.c* and *dec.c*. We modify Fisher's programs into centered BFT with quadtrees and compare our results with Fisher's. The programs were compiled by Watcom C/C++32 version 10.0 under MS-DOS 6.2 environment and run on a 486-66 Mhz PC. Five test data sets, Lena, Clown, Camera, Hamlet and Kgirl, were used in our experiments. The data sets are 256-graylevel images of size 256 pixels by 256 pixels and ftp'able from *eedsp.gatech.edu:/database/images* directory. We first show the results of Fisher's original programs as the benchmarks in section 6.1. The results of the centered Fisher's model (centered order-0 BFT with quadtrees) are given in section 6.2. The results of the centered Fisher's model with new beta quantization are presented in section 6.3. We compare the results of the above three sections in section 6.4. The results of the centered BFT with quadtrees and re-centered BFT with quadtrees are presented in section 6.5 which includes: the results of re-centered order-2 BFT/no-cross-term with quadtrees, re-centered order-2 BFT/cross-term with quadtrees, re-centered order-3 BFT/no-cross-term with quadtrees, and re-centered order-3 BFT/cross-term with quadtrees.

6.1 The benchmarks: Fisher's original model

The default options of the encoding program are used in our experiments. The tolerance criterion (e_c, in the pseudo-code of section 5.1) for fidelity is 8.0, i.e., the acceptable RMS error between the range block and the corresponding domain block should be less than 8.0. The width and height of the input image are 256 pixels. The minimum quadtree depth is 4, i.e., the range-domain matching process will begin after four quadtree partitions have been done and the width and height of the range block will be $256 / 2^4 = 16$. The maximum quadtree depth is 6, i.e., the range will not be further partitioned when its width and height are of size $256 / 2^6 = 4$. Five bits are used to quantize α_i. Seven bits are used to quantize β_i. Only domains with the same or near classification are compared with the range in order to reduce the number of domain-range comparisons.

The decoding program uses an all-black image (255 graylevel for each pixel value) as the initial image from which the partitioned iterated function is iterated. This initial image size is the same as the encoded image size, 256×256. Postprocessing is used to smooth the boundaries between two range blocks. This not only alleviates the blocky artifacts practically, but also reduces the overall RMS error.

The file size of every input image is 65,536 bytes. The results of the compression are shown in Table 6.1. Table 6.1 shows a considerable variation in the compression results. Hamlet is very well compressed because the background is plainly gray, while Clown is the hardest. The compressed size of Hamlet is only 40% of that of Clown. The decompressed image

TABLE 6.1

COMPRESSION RESULTS OF FISHER'S PIFS WITH QUADTREES

	Encoded size (bytes)	*No. of transforms*	*Decoded RMS*
Lena	7197	2296	8.88
Clown	9366	2983	7.73
Camera	6222	1996	11.11
Hamlet	3822	1243	4.64
Kgirl	6191	1981	6.69

fidelity also varies significantly. Hamlet converges to the smallest RMS error (4.64), while Camera converges to a high RMS error (11.11) which is higher than the preselected e_c value (8.0). The RMS error of Hamlet is only 42% of Camera's.

The original images and Fisher's decompressed images are listed in Figure 6.1 for comparison.

The RMS errors at the end of each decoded iteration for those test images are shown in Table 6.2. The attractor images converge in 7 to 9 iterations. Usually, those images with smaller decompressed RMS errors converge faster, such as Hamlet, Kgirl and Clown.

6.2 Centered Fisher's model

In Jacquin's original PIFS model, the α_i and β_i are correlated. We decorrelate α_i and β_i by centering the R_i and D_i. The compression results of our centered PIFS model are shown in Table 6.3.

256×256 Original Images

Origin Lena

Original Clown

Original Camera

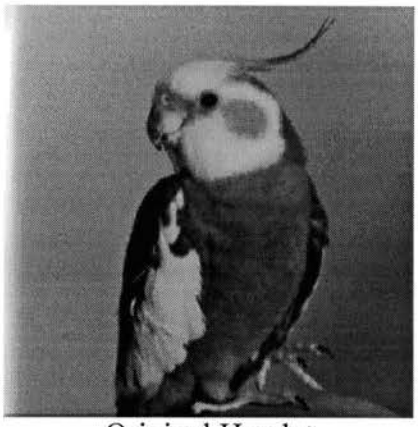
Decompressed Images by Fisher's PIFS

Decompressed Lena, RMS = 8.88

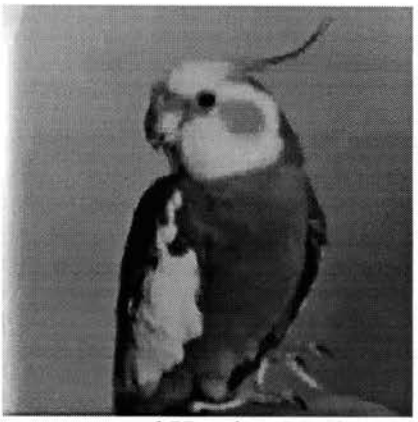
Decompressed Clown, RMS = 7.73

Decompressed Camera, RMS = 11.11

Figure 6.1 List of original images and Fisher's decompressed images.



Original Hamlet



Decompressed Hamlet, RMS = 4.64

Original Kgirl

Decompressed Kgirl, RMS = 6.69

Figure 6.1 List of original images and Fisher's decompressed images. (continued)

TABLE 6.2

THE RMS ERROR OF DECOMPRESSION SEQUENCES
USING FISHER'S PIFS WITH QUADTREES

	Lena	*Clown*	*Camera*	*Hamlet*	*Kgirl*
1st iteration RMS	62.19	58.14	60.91	49.59	39.60
2nd iteration RMS	35.95	32.88	33.69	23.62	22.62
3rd iteration RMS	21.89	19.58	21.02	12.09	13.99
4th iteration RMS	12.28	10.03	13.52	6.35	8.30
5th iteration RMS	9.48	8.04	11.61	4.81	6.90
6th iteration RMS	8.97	7.76	11.19	4.66	6.70
7th iteration RMS	8.90	7.73	11.13	4.64	6.69
8th iteration RMS	8.89	7.73	11.12	4.64	6.69
9th iteration RMS	8.88	7.73	11.11	4.64	6.69

TABLE 6.3

COMPRESSION RESULTS OF CENTERED PIFS WITH QUADTREES

	Encoded size (bytes)	*No. of transforms*	*Decoded RMS*
Lena	7185	2293	8.87
Clown	9356	2980	7.73
Camera	6232	1999	11.12
Hamlet	3819	1243	4.62
Kgirl	6188	1981	6.66

Comparing Table 6.1 with Table 6.3, most images are compressed to smaller size by this centering method, except Camera which is 10 bytes more. The RMS errors of most attractor images are slightly reduced, except Camera which is 0.01 more. Roughly speaking, the centering method may slightly improve the compression ratio and the decompressed image fidelity.

The principal advantage of this centering method is the faster convergence of the decompression process. The RMS errors at the end of each decompression iteration for those test images are shown in Table 6.4.

TABLE 6.4

THE RMS ERROR OF DECOMPRESSION SEQUENCES
USING CENTERED PIFS WITH QUADTREES

	Lena	*Clown*	*Camera*	*Hamlet*	*Kgirl*
1st iteration RMS	15.74	17.61	19.71	9.84	12.17
2nd iteration RMS	11.33	11.45	14.32	6.48	8.37
3rd iteration RMS	8.98	7.97	11.14	4.94	6.80
4th iteration RMS	8.88	7.73	11.12	4.64	6.66
5th iteration RMS	8.87	7.73	11.11	4.63	6.66
6th iteration RMS	8.87	7.72	11.12	4.62	6.66
7th iteration RMS	8.87	7.73	11.12	4.62	6.66
8th iteration RMS	8.87	7.73	11.12	4.62	6.66
9th iteration RMS	8.87	7.73	11.12	4.62	6.66

The decompression converges to the attractor image in 4 to 6 iterations for most images. The decompression sequences of Lena using PIFS and centered PIFS methods have been listed in Figure 4.8. The decompression sequences of Clown, Camera, Hamlet and Kgirl using PIFS and centered PIFS methods are in Appendix A, B, C and D respectively. The comparison curves of the convergent speed for each image also have been listed in Figures 4.9 to 4.13. Obviously, the major advantage of the centering is the speedy convergence for decompression without sacrificing the compression ratio and the decompressed image fidelity. In fact, the compression ratio and the decompressed image fidelity are even slightly improved in average for our testing images.

6.3 Centered Fisher's model with new β' quantization

The original Fisher's model used the following method to quantize the β_i value into a 7-bit integer.

$$\text{if } (\alpha_i > 0) \text{ then } \beta_i = \beta_i + \alpha_i \cdot 255$$

$$\text{quantized\_}\beta_i = \left\lfloor 0.5 + \frac{\beta_i}{\left((1+|\alpha_i|)\cdot 255\right)} \cdot 127 \right\rfloor . \tag{6.1}$$

Fisher utilizes the sign information of α_i to quantize β_i efficiently, because α_i and β_i are correlated. Now, the centered model decorrelates the α_i and β_i, so a regular quantization for the new β_i (i.e., β_i') would be more appropriate. The β_i' is actually $\bar{r}_i$ (Equation 4.22), i.e., the value of β_i' is from 0 to 255. The objective is to use 7 bits to represent β_i', hence

$$\text{quantized\_}\beta_i' = \left\lfloor 0.5 + \frac{\beta_i'}{255} \cdot 127 \right\rfloor . \tag{6.2}$$

The compression results of the centered Fisher's model with new β_i' quantization are shown in Table 6.5.

TABLE 6.5

COMPRESSION RESULTS Of CENTERED PIFS WITH QUADTREES AND NEW β_i' QUANTIZATION

	Encoded size (bytes)	*No. of transforms*	*Decoded RMS*
Lena	7198	2296	8.85
Clown	9355	2977	7.71
Camera	6243	1996	11.11
Hamlet	3825	1243	4.61
Kgirl	6152	1969	6.69

Let us compare Table 6.3 and Table 6.5. Some images are better compressed, while some images are not. The Kgirl image size is decreased the most, 36 bytes. The RMS errors of most attractor images are slightly reduced, except Kgirl which is 0.03 more. The slight decrease of Kgirl's image fidelity is the tradeoff of the 36 bytes saving of the compressed file size. However, 0.03 RMS difference on the decompressed image is not perceivable by human eyes. Roughly speaking, the new β_i' quantization method may further slightly improve the decompressed image fidelity or the compression ratio compared with the results of the centered PIFS with the old β_i quantization.

Because the centering model is also adopted by the new β_i' quantization method, it enjoys the same advantage of a faster convergence of decompression process. The RMS errors at the end of each decompression iteration for those test images are shown in Table 6.6.

TABLE 6.6

THE RMS ERROR OF DECOMPRESSION SEQUENCES
USING CENTERED PIFS WITH QUADTREES AND NEW β_i' QUANTIZATION

	Lena	*Clown*	*Camera*	*Hamlet*	*Kgirl*
1st iteration RMS	15.68	17.59	19.70	9.82	12.20
2nd iteration RMS	11.31	11.43	14.31	6.48	8.40
3rd iteration RMS	8.95	7.96	11.13	4.93	6.83
4th iteration RMS	8.86	7.72	11.11	4.63	6.70
5th iteration RMS	8.85	7.71	11.11	4.61	6.70
6th iteration RMS	8.85	7.71	11.11	4.61	6.69
7th iteration RMS	8.85	7.71	11.11	4.61	6.69
8th iteration RMS	8.85	7.71	11.11	4.61	6.69
9th iteration RMS	8.85	7.71	11.11	4.61	6.69

Similar to the results of the centered PIFS with the old β_i quantization in Table 6.4, the decompression converges to the attractor image in 4 to 6 iterations. This new β_i' quantization does not further improve the convergent speed.

6.4 Comparison of Fisher's model, Centering and new β_i' Quantization

The compression results of the Lena image in Table 6.1 to 6.6 are summarized in Table 6.7 for comparison. The results are similar for the other test images.

TABLE 6.7

LENA COMPRESSION RESULTS OF PIFS MODEL, CENTERED PIFS AND CENTERED PIFS WITH NEW β_i' QUANTIZATION

Methods *Compressed size*	*Fisher* *7197 bytes*	*Centered* *7185 bytes*	*β_i' Quantization* *7198 bytes*
1st iteration RMS	62.19	15.74	15.68
2nd iteration RMS	35.95	11.33	11.31
3rd iteration RMS	21.89	8.98	8.95
4th iteration RMS	12.28	8.88	8.86
5th iteration RMS	9.48	8.87	8.85
6th iteration RMS	8.97	8.87	8.85
7th iteration RMS	8.90	8.87	8.85
8th iteration RMS	8.89	8.87	8.85
9th iteration RMS	8.88	8.87	8.85

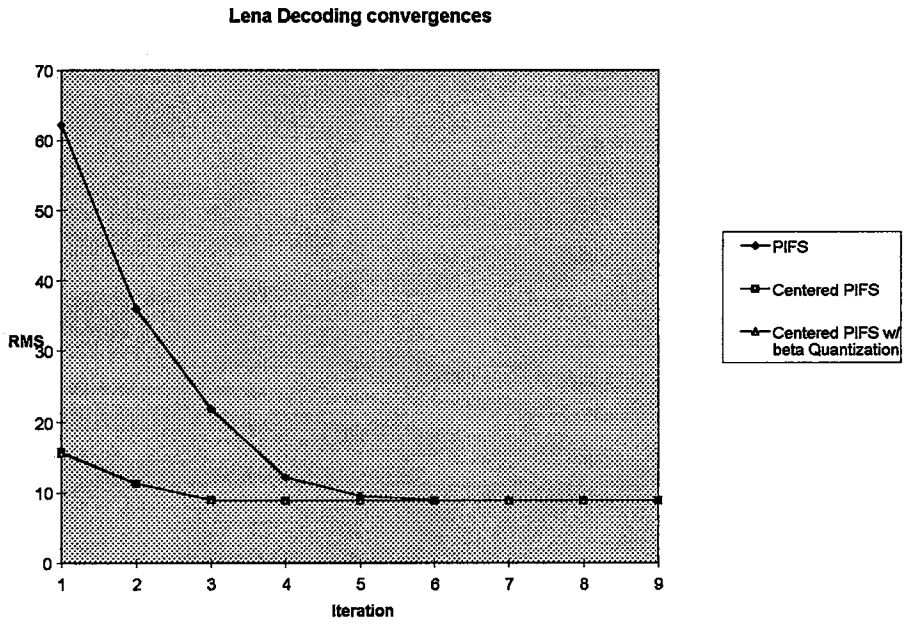
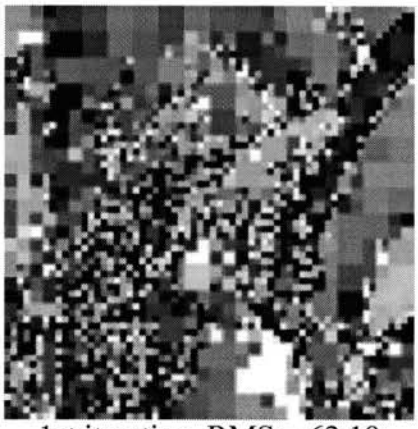


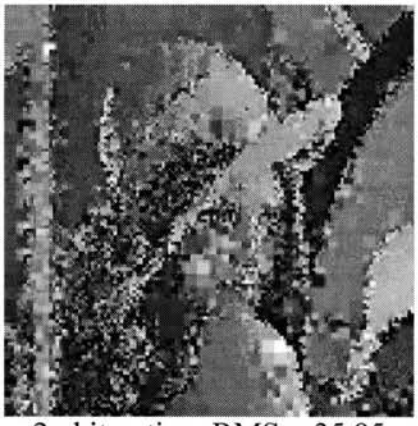
Figure 6.2 The comparison of decompression convergences of three models.

Figure 6.2 shows the decoding convergence of the above three methods. Both centered and centered-with-β'-quantization methods converge to the fixed point at the fifth iteration, while Fisher's method converges at the ninth iteration. The new methods converge faster, but also converge to a slightly smaller RMS error. Due to the fact that β_i' is more precisely quantized, the β_i' quantization method converges to the lowest RMS, 8.85, at the expense of a worse compression ratio, 14 bytes larger than the simple centered method. Their RMS errors at the first iteration are less than at the third iteration of Fisher's method, i.e., their resulting image at the first iteration looks better than the resulting image at the third iteration of Fisher's method. The decompression image sequences of Lena using primitive PIFS's and our centered PIFS with new β_i' quantization methods are shown in Figure 6.3.

Fisher's PIFS



1st iteration, RMS = 62.19



2nd iteration, RMS = 35.95

3rd iteration, RMS = 21.89

Centered PIFS with new β_i' quantization

1sr iteration, RMS = 15.68

2nd iteration, RMS = 11.31

3rd iteration, RMS = 8.95

Figure 6.3 Decompression sequences of Lena using PIFS and centered PIFS with new β_i' quantization.

4th iteration, RMS = 12.28

4th iteration, RMS = 8.86

5th iteration, RMS = 9.48

5th iteration, RMS = 8.85

6th iteration, RMS = 8.97

6th iteration, RMS = 8.85

Figure 6.3 Decompression sequences of Lena using PIFS and centered PIFS with new β_i' quantization. (continued)

7th iteration, RMS = 8.90

7th iteration, RMS = 8.85

8th iteration, RMS = 8.89

8th iteration, RMS = 8.85

9th iteration, RMS = 8.88

9th iteration, RMS = 8.85

Let us investigate the decoded images of the centered method with new β_i' quantization. The decoded image at the first iteration is actually composed of the range blocks with flat mean value. This can be easily verified from Equation 4.22:

$$r_i \leftarrow \alpha_i \cdot (d_i - \bar{d}_i) + \beta_i', \text{ where } \beta_i' = \bar{r}_i. \tag{6.3}$$

The $\bar{r}_i$ is our β_i'. Since the initial image is all-black, every pixel value is zero. The d_i is zero, so is $\bar{d}_i$. Hence, the α_i term does not contribute any value in the first iteration. The resulting r_i values in a range block will have the same value, $\bar{r}_i$. Due to the decorrelation of α_i and β_i by the centering, the new β_i' is actually $\bar{r}_i$ which is much meaningful than the old β_i. This explains why the decoded image at the first iteration looks much better than that from Fisher's model.

The decoded image at the 2nd iteration looks almost the same as the final attractor image. The decoded image at the 3rd iteration is indistinguishable from the final attractor image by human eyes.

6.5 Centered BFT with Quadtrees

Fisher's quadtree partition model will partition a range block into four sub-range blocks when there is no "good enough" domain block that can be contractively mapped onto the range block. Here, "good enough" means the image above a domain block can be transformed into a similar image above the range block, i.e., the RMS error between the two image blocks should be less than a preselected threshold. We propose a method that provides the range block a second chance, before being further partitioned, to find its corresponding domain block. The best-but-not-good-enough domain block is tested on

higher order BFT transformations to find out whether a good BFT transformation exists to make that domain block similar enough to the range block. The test proceeds from an order-1 BFT until a good transformation is found or a maximum order is reached. If a good transformation is found, the coefficients are recorded as the representation of the range block. If even the maximum-order BFT cannot transform that domain block into the range block, the range block is then further partitioned. The detailed algorithm is given in section 5.6.4.

6.5.1 Re-centering the polynomial

Different maximum orders of BFT were implemented based on the above algorithm. We not only used centered range and domain blocks, but also re-centered the polynomial after all coefficients were quantized (section 5.6.3). The re-centering adjusts the β_i' value to make the integral of the polynomial terms zero. The benefit of this adjustment is a better compression ratio without reducing the image fidelity. We set the maximum order of the BFT to 2 for comparing the effect of re-centering. The experimental results are presented in Table 6.8. The maximum order of the BFT is 2, which means that order-1 BFT is first tried to transform the domain block into the range block; if that fails, order-2 BFT is then tried before the range is further partitioned. According to the practical distribution of all transformation coefficients, the coefficients of the x term and the y term are limited between -8 and 8 and quantized into 5 bits. Any other values out of this range will be truncated. The coefficients of quadratic terms and above are limited between -1 and 1 and quantized into 5 bits.

TABLE 6.8

THE EFFECT OF RE-CENTERING POLYNOMIAL ON ORDER-1,2 BFT

Method	*Centered Range & Domain Order-1, 2 BFT*		*Centered Range & Domain Order-1, 2 BFT Re-centering Polynomial*			
	Encoded size (Bytes)	*Decoded RMS*	*Encoded size (Bytes)*	*Difference (Bytes)*	*Decoded RMS*	*Difference RMS*
Lena	6622	9.00	6595	-27	9.00	0
Clown	8903	7.95	8769	-134	8.01	+0.06
Camera	6197	11.13	6197	0	11.13	0
Hamlet	3403	4.96	3355	-48	5.01	-0.05
Kgirl	5384	6.97	5354	-30	6.98	-0.01

The compression ratios are improved by the re-centering polynomial with very little or no cost of decompressed image fidelity for most images, and with no effect on Lena or Camera at all. The compressed size of the Clown image is reduced the most, 134 bytes, with the highest cost of degradation of decompressed image fidelity, 0.06, which is not distinguishable by human eyes.

Because the re-centering polynomial producing a better compression ratio, we adopted this modification in all of the following experiments (section 6.5.2).

6.5.2 Higher order BFTs

We implemented some higher order BFTs with quadtrees to see the effect on the compression ratio. The domain and range blocks were centered and the polynomial was re-centered too. We also illustrate the effect of cross terms (xy, x^2y and xy^2) in the polynomial by setting the maximum order of the BFT to 3. In Table 6.9, we list the compression ratio and the decompressed image fidelity achieved by Fisher's original model in column 1. Column 2 gives the results of our model which tries to transform the domain with order-1 BFT first, then adds order-2 terms. Column 3 gives the results of our model which tries to transform the domain with order-1 BFT first, then adds order-2 terms, then adds order-3 terms. Column 4 gives the results of our model which tries to transform the domain with order-1 BFT first, then adds order-2 terms, then adds order-3 terms, and then adds the 3rd order cross-terms x^2y and xy^2.

The order-1,2,3,x^2y&xy^2 method (Column 4) has the best compression ratio at a cost of slightly reduced decompressed image fidelity. We notice that the additional order-

3 terms improve the compression ratio much less than the order-1 and order-2 terms do. We also notice that the 3rd-order cross-terms, x^2y and xy^2, do not help much in the compression ratio and the cost of reduced image fidelity is higher than for the 3rd order terms.

There is no 2nd-order cross-term xy at all in our previous experiments (Table 6.9). The effect of an xy term in higher order BFT is illustrated in Table 6.10. Columns 1 and 2 are the same as in Table 6.9. Column 3 gives the results of our model which tries to transform the domain with order-1 BFT first, then adds order-2 terms, and then adds an xy term. Column 4 gives the results of our model which tries to transform the domain with order-1 BFT first, then adds order-2 terms, then adds an xy term, and then adds order-3 terms. Column 5 gives the results of our model which tries to transform the domain with order-1 BFT first, then adds order-2 terms, then adds an xy term, then adds order-3 terms, and then adds x^2y and xy^2 terms. We notice that the additional xy term in column 3 improves the compression ratio quite a bit, although only one extra coefficient was added.

The decoded images for Fisher's original method (Column 1) and our order-1,2,xy,3,x^2y&xy^2 method (Column 5) are presented in Figure 6.4 for comparison.

TABLE 6.9

RESULTS OF HIGHER ORDER BFTs
WITHOUT THE 2ND ORDER CROSS-TERM XY

Method	*Fisher's*		*Order-1,2*		*Order-1,2,3*		*Order-1,2,3,x^2y&xy^2*	
	Encoded Size (Bytes)	*Decoded RMS*	*Encoded Size (Bytes)*	*Decoded RMS*	*Encoded Size (Bytes)*	*Decoded RMS*	*Encoded Size (Bytes)*	*Decoded RMS*
Lena	7197	8.88	6595	9.00	6491	9.03	6423	9.10
Clown	9366	7.73	8769	8.01	8483	8.12	8338	8.34
Camera	6222	11.11	6197	11.13	6119	11.15	6103	11.17
Hamlet	3822	4.64	3355	5.01	3327	5.03	3245	5.30
Kgirl	6191	6.69	5354	6.98	5263	7.02	5158	7.25

TABLE 6.10

RESULTS OF HIGHER ORDER BFTS
WITH THE 2ND ORDER CROSS-TERM XY

Methods	*Fisher's*		*Order-1,2*		*Order-1,2,xy*		*Order-1,2,xy,3*		*Order-1,2,xy,3,x^2y&xy^2*	
	Encoded Size (Bytes)	*Decoded RMS*	*Encoded Size (Bytes)*	*Decoded RMS*	*Encoded Size (Bytes)*	*Decoded RMS*	*Encoded Size (Bytes)*	*Decoded RMS*	*Encoded Size (Bytes)*	*Decoded RMS*
Lena	7197	8.88	6595	9.00	6412	9.08	6336	9.09	6280	9.15
Clown	9366	7.73	8769	8.01	8561	8.09	8346	8.18	8250	8.35
Camera	6222	11.11	6197	11.13	6149	11.14	6084	11.16	6078	11.18
Hamlet	3822	4.64	3355	5.01	3210	5.13	3162	5.16	3115	5.41
Kgirl	6191	6.69	5354	6.98	5186	7.05	5068	7.11	4966	7.43

Decoded Images by Fisher's PIFS

Fisher's Decoded Lena
RMS = 8.88

Fisher's Decoded Clown
RMS = 7.73

Fisher's Decoded Camera
RMS = 11.11

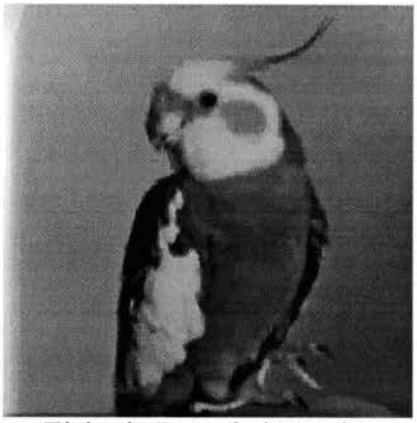
Decoded Images by Order-3 BFT

Order-3 BFT Decoded Lena
RMS = 9.15

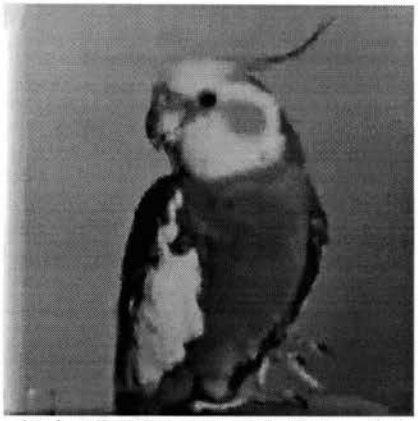
Order-3 BFT Decoded Clown
RMS = 8.35

Order-3 BFT Decoded Camera
RMS = 11.18

Figure 6.4 List of Fisher's decoded images and order-3 BFT decoded Images.



Fisher's Decoded Hamlet
RMS = 4.64



Order-3 BFT Decoded Hamlet
RMS = 5.41

Fisher's Decoded Kgirl
RMS = 6.69

Order-3 BFT Decoded Kgirl
RMS = 7.43

Figure 6.4 List of Fisher's decoded images and order-3 BFT decoded Images.

(continued)

CHAPTER VII

SUMMARY AND CONCLUSIONS

7.1 Epilogue

This dissertation presents a better fractal image compression model with two innovations: *centering* and *BFT with quadtrees*.

The *centering* method reduces the decompression time to about half of the traditional model's time and the contents of the decompressed image are recognizable as early as the end of the first decompression iteration. With this advantage, the centering method does not sacrifice the compression ratio and the decompressed image fidelity. In fact, the compression ratio and the decompressed image fidelity are even slightly improved in average for our testing images.

The *BFT with quadtrees* model can compress the image into a smaller file size at the expense of a slightly worse decompressed image fidelity. However, the slight deterioration is usually not perceivable by human eyes.

The purpose of the centering method is to decorrelate the coefficients of the contractive mappings. The decorrelated coefficients are much more meaningful than the coefficients of the original model. For example, the coefficients (α_i and β_i) of the PIFS model are highly correlated. After decorrelation, β_i' represents the mean value of the range block.

The main effect of decorrelation is a faster decompression speed. At the end of the first decompression iteration, the centered PIFS obtains a much better image than the

original PIFS model does, because the mean value of each range block is recovered from β_i'. Hence the contents of the decompressed image are recognizable at the end of the first decompression iteration. This advantage also leads to a faster convergence of the centered PIFS model. The centering method is not only effective for the PIFS model, but also applicable to most fractal image compression models with range blocks and domain blocks.

The PIFS is actually an order-0 BFT. The reason that Monro extended PIFS into higher order BFTs was to reduce or eliminate the searching time for pairing. However, Monro's intuition about local self-similarity was proven wrong by Fisher. The successfully matched domains are essentially random and there is no preference for local domains. Searching is also necessary for higher order BFTs in order to find good matched domains.

Fisher developed a program for *fractal image compression with quadtrees* based on PIFS. When the searching fails to find a good enough domain block for a range block, that range block is further partitioned into four smaller range blocks. Each new range block will be compressed individually, hence the searching process is repeated all over again.

Not only is the repeated searching very time consuming, but also the compression ratio becomes worse because four more sets of coefficients will be recorded in the compressed file. In order to reduce the repeated searching and improve the compression ratio, we proposed a hybrid model (*BFT with quadtrees*) that combines Fisher's model and Monro's model.

When the best match domain block is still not acceptable, higher order BFTs are applied to this best match domain block until the image fidelity is satisfied or the maximum order of BFT is reached. If even the maximum order BFT still cannot make a satisfactory pairing, the range block is then partitioned into several smaller blocks which should make it easier to find a domain block that will achieve the target image fidelity.

This strategy eliminates the necessary searching process for higher order BFTs, because the best domain block for PIFS (order-0 BFT) is ready for use. It also reduces the frequency of partitioning range blocks. Fewer partitioned range blocks means less repeated searching and less storage for coefficients of contractive mappings. Hence, a better compression performance and a better compression ratio are achieved at the cost of slightly worse decompressed image fidelity.

We combine the two innovations into a model, *centered BFT with quadtrees*, which enjoys the advantages from both innovations. Compared with Fisher's original model, this model can compress an image into a smaller file size with only slight degradation of the decompressed image fidelity. The convergence of decompression is fast too, because this model utilizes the centering method.

The centering of the higher order BFTs is a little bit different from the centering of PIFS (order-0 BFT), because the quantized coefficients may bring up some bias. We propose a re-centering method to avoid the bias: the new β_i' is derived by re-centering the polynomial with quantized coefficients. The compression ratio is improved by the re-centering with very little or no deterioration in decompressed image fidelity.

We implemented some different maximum orders of BFTs with quadtrees to see the effect on the compression ratio. The results show that higher maximum orders will produce a better compression ratio and slightly worse decompressed image fidelity. If there is no xy term, the 3rd order cross-terms, x^2y and xy^2, do not help much on the compression ratio and the degradation of image fidelity is worse than for the 3rd order terms (see Table 6.9). We also notice that the additional xy term reduces the compressed file size notably although only one extra coefficient is added (see Table 6.10).

7.2 Contributions and Future Research

The major contribution of the *centered BFT with quadtrees* model developed by this work is to provide a fractal image compression technique that can compress better and decompress faster with only slightly degradation of image fidelity. We outline the advantages as follows:

(1) Centering can make the decompression faster. The decompressed image at the end of the first decompression iteration is composed of mean-value range blocks, hence the image fidelity is much better than that of the original PIFS model. The contents of the image are recognizable at the end of the first decompression iteration. Furthermore, the image at the end of the second iteration is almost indistinguishable from the final attractor image. In addition, centering, by itself, does not degrade the image fidelity. In fact, centering slightly improves the compression ratio and the decompressed image fidelity in average for our testing images.

(2) The model *BFT with quadtrees* provides a better compression ratio with a negligible degradation of decompressed image fidelity.

(3) The combined model *centered BFT with quadtrees* enjoys advantages from both methods: better compression ratio and faster decompression speed at the cost of slightly worse decompressed image fidelity.

(4) Re-centering the polynomial further improves the compression ratio.

(5) If the maximum order of the BFT is higher, a better compression and a slight degradation of decompressed image fidelity will be obtained. The xy term plays an important role for a better compression.

(6) Both *centering* and *BFT with quadtrees* can be applied to most fractal image compression models.

Future research directions include:

1. Adaptive quantization of coefficients

The coefficients a_1, b_1, a_2, b_2, c_{11}, ... derived from the solution of the system of equations are floating numbers. They are quantized on a uniform scale before actually being stored as the representation of the compressed image block. In practice, they are usually distributed with a high concentration of zero values. We may more precisely describe the compressed image block if a Lloyd-Max quantizer [Lloy82] or others are applied. Hence, a better image fidelity might be obtained without decreasing the compression ratio.

However, we will not know their distributions before the end of the compression. A carefully designed adaptive quantizer might solve this problem.

2. Video compression

Video displays many frames in a second, so a fast decompression scheme is necessary. The centering method reduces the decompression time to about half of the traditional models' time, hence it can be a good alternative for video decompression. Besides, the image quality of TV is not extremely high, so a fractal image compression technique that can tradeoff the image quality with compression ratio might do the job. After two iterations, the centering method provides an almost indistinguishable image from the final attractor image. Video decompression using the centering method can also tradeoff the decompression time and image fidelity according to its need.

Bibliography

[AdPo94] Adler, M., Popp, H., Introduction to MPEG, in: *FAQ of comp.compression*, ftp://rtfm.mit.edu:/pub/usenet/news.answers/compression-faq/part2, 1995

[BaMaKa93] Baharav, Z., Malah, D., Karnin, E. D., *Hierarchical interpretation of fractal image coding and its applications to fast decoding*, in: International Conference on Digital Signal Processing, Cyprus, July 1993.

[Bani94] Bani-Eqbal, B., *Speeding up fractal image compression*, Report, Department of Computer Science, University of Manchester, UK, Sep. 1994.

[BEHL86] Barnsley, M. F., Ervin, V., Hardin, D., Lancaster, J., *Solution of an inverse problem for fractals and other sets*, Proc. Natl. Acad. Sci. USA 83 (1986) 1975-1977.

[Barn88] Barnsley, M., *Fractal modelling of real world images*, in: *Fractal Images*, H.-O. Peitgen and D. Saupe (ed.), Springer-Verlag, New York, 1988, p.238.

[Barn89] Barnsley, M., *Fractals Everywhere*, Academic Press, San Diego, 1989.

[Barn93] Barnsley, M., *Fractals Everywhere*, 2nd Ed., Academic Press, San Diego, 1993.

[BaHu92] Barnsley, M., Hurd, L., *Fractal Image Compression*, AK Peters, Wellesley, 1992.

[BaSl87] Barnsley, M. F., Sloan, A. D., *Chaotic compression*, Computer Graphics World, Nov. 1987.

[BaSl88]Barnsley, M. F., Sloan, A. D., *A better way to compress images*, BYTE Magazine, Jan. 1988.

[BaSl90] Barnsley, M. F., Sloan, A., *Methods and apparatus for image compression by iterated function system*, United States Patent #4,941,193.

[BaSl91] Barnsley, M. F., Sloan, A., *Method and apparatus for processing digital data*, United States Patent #5,065,447.

[BaVo94] Barthel, K. U., Voye, T., *Adaptive fractal image coding in the frequency domain*, in: *Proceedings of International Workshop on Image Processing*, Budapest, June 1994.

[Bell90] Bell, T. C., *Text Compression*, Prentice-Hall, New Jersey, 1990

[Box60] Box, G. E. P., *Fitting empirical data*, Ann. N. Y. Acad. Sci., 86 (1960) 792-816.

[Burt95] Burt, J., *Archive Comparison Table*, ftp://wuarchive.wustl.edu:/pub/MSDOS_UPLOADS/archivers/act-25.zip, May 30, 1995

[ChSh91] Cheung, K.-M., Shahshahani, M., *A comparison of the fractal and JPEG algorithms,* TDA Progress Report 42,107 (1991) 21-26.

[CRDMREK94] Cicconi, P., Reusens, E., Dufaux, F., Moccagatta, I., Rouchouze, B., Ebrahimi, T., Kunt, M., *New trends in image data compression,* Computerized Medical Imaging and Graphics, 18, 2 (1994), 107-124.

[DaCh94] Davoine, F., Chassery, J.-M., *Adaptive delaunay triangulation for attractor image coding,* in: *12th International Conference on Pattern Recognition,* Jerusalem, Oct. 1994.

[DrSm66] Drapper, N., Smith, H., *Applied Regression Analysis*, John Wiley & Sons, Inc., New York, 1966.

[Dudb94] Dudbridge, F., *Fast image coding by a hierarchical fractal construction,* Institute for Nonlinear Science, University of California, San Diego, La Jolla, CA 92093-0402, 1994.

[FGHS94] Frigaard, C., Gade, J., Hemmingsen, T. T., Sand, T., *Image compression based on a fractal theory,* Report, Institute for Electronic Systems, Aalborg University, Denmark, 1994.

[FiJaBo91] Fisher, Y., Jacobs, E. W., Boss, R. D., *Fractal image compression using iterated transforms.* Technical Report 1408, Naval Ocean Systems Center, San Diego, CA, 1991.

[FiMe94] Fisher, Y., Menlove, S., *Fractal encoding with HV partitions,* in: *Fractal Encoding -- Theory and Applications to Digital Images*, Y. Fisher (ed.), Springer-Verlag, New York, 1994.

[FiRoSh94a] Fisher, Y., Rogovin, D., Shen, T. P., *A comparison of fractal methods with DCT and Wavelets,* Report, Institute for Nonlinear Science, University of California, San Diego, La Jolla, CA 92093-0402, 1994.

[FiRoSh94b] Fisher, Y., Rogovin, D., Shen, T. P., *Fractal (Self-VQ) encoding of video sequences,* Proceedings of the SPIE, Visual Communications and Image Processing '94 (VCIP'94), Chicago, IL, Sep. 1994.

[Fish92] Fisher, Y., *Fractal image compression,* SIGGRAPH '92 Course Notes, Vol. 12, 7.1-7.19. 1992.

[Fish95] Fisher, Y. (ed.), *Fractal image compression -- Theory and Application*, Springer-Verlag, New York, 1995.

[FoVr94a] Forte, B., Vrscay, E. R., *Solving the inverse problem for function/image approximation using iterated function systems, I. Theoretical basis,* Fractals 2,3 (1994) 325-334.

[FoVr94b] Forte, B., Vrscay, E. R., *Solving the inverse problem for function/image approximation using iterated function systems, II. Algorithm and computations,* Fractals 2,3 (1994) 325-334.

[Gall91] Gall, D. L., *MPEG: A Video Compression Standard for Multimedia Applications*, Communications of the ACM, 34,4 (1991) 46-58.

[GhHu93] Gharavi-Alkhansari, M., Huang, T., *A fractal-based image block-coding algorithm,* Proc. ICASSP 5 (1993) 345-348.

[Gilg91] Gilge, M., *Distortion accumulation in image transform coding/decoding cascades,* Technical Report TR-91-064, International Computer Science Institute, Berkeley, CA 94704, Dec. 1991.

[GoOr] Golub, G. H., Ortega, J. M., *Scientific Computing and Differential Equations,* Academic Press, Boston.

[GoVa89] Golub, G. H., Van Loan, C. F., *Matrix Computations,* 2nd Ed., The Johns Hopkins University Press, Baltimore, 1989.

[HiJaSe94] Hilton, M. L., Jawerth, B. D., Sengupta, A., Compressing Still and Moving Images with Wavelets, Multimedia Systems, 2, 3 (1994).

[Holl91] Hollatz, S.A., *Digital Image Compression with Two-dimensional Affine Fractal Interpolation Functions*, Report, Department of Mathematics and Statistics, University of Minnesota-Duluth, March 1991.

[Hutc81] Hutchinson, J., *Fractals and self-similarity,* Indiana University Journal of Mathematics 30 (1981) 713-747.

[JaBoFi89] Jacobs, E. W., Boss, R. D., Fisher, Y., *Fractal based image compression II,* Technical Report 1362, Naval Ocean Systems Center, San Diego, CA 92152-5000, 1989.

[JaFiBo92] Jacobs, E. W., Fisher, Y., Boss, R. D., *Image compression: A study of the iterated transform method,* Signal Processing 29 (1992) 251-263.

[Jacq89] Jacquin, A. E., *A fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding.* Ph.D. Thesis, Georgia Institute of Technology, August 1989.

[Jacq90a] Jacquin, A. E., *Fractal image coding based on a theory of iterated contractive image transformations,* Proc. SPIE's Visual Communications and Image Processing (1990) 227-239.

[Jacq90b] Jacquin, A. E., *A novel fractal block-coding technique for digital images,* Proc. ICASSP 4 (1990) 2225-2228.

[Jacq92] Jacquin, A. E., *Image coding based on a fractal theory of iterated contractive image transformations,* IEEE Trans. Image Processing 1 (1992) 18-30.

[Jacq93] Jacquin, A. E., *Fractal image coding: A review,* Proceedings of the IEEE 81, 10 (1993) 1451-1465.

[Kaou92] Kaouri, A. E., *Fractal coding of still images,* in: *IEEE 6th International Conference on Digital Processing of Signals in Communications,* 1991.

[Kocs89] Kocsis, S., *Digital compression and iterated function systems,* SPIE-Application of Digital Image Processing 1153 (1989) 19-27.

[Komi94] Kominek, J., *Still image compression -- An issue of quality,* Report, Department of Computer Science, University of Waterloo, 1994.

[Komi95] Kominek, J., Introduction to Fractal Compression, FAQ of comp.compression, ftp://rtfm.mit.edu:/pub/usenet/news.answers/comperssion-faq/part2, 1995

[Lane95] Lane, T., *Introduction to JPEG,* in: FAQ of JPEG, ftp://rtfm.mit.edu:/pub/usenet/news.answers/compression-faq/part1, part2, 1995.

[LeØiRa93] Lepsøy, S., Øien, G. E., Ramstad, T., *Attractor image compression with a fast non-iterative decoding algorithm,* Proc. ICASSP 5 (1993) 337-340.

[Lloy82] Lloyd, S., *Least squares quantization in PCM,* IEEE Transactions on Information Theory, IT-28, 2 March 1982.

[Mand77] Mandelbrot, B., *The Fractal Geometry of Nature,* New York, (updated and augmented copy) 1983.

[MeKi67] Mezaki, R., Kittrell, J. R., *Parametric sensitivity in fitting nonlinear kinetic models,* Industrial and Engineering Chemistry 59 (1967) 63-69.

[Monr93] Monro, D. M., *A hybrid fractal transform,* Proc. ICASSP 5 (1993) 169-172.

[MoDu92a] Monro, D. M., Dudbridge, F., *Fractal approximation of image blocks,* Proc. ICASSP 3 (1992) 485-488.

[MoDu92b] Monro, D. M., Dudbridge, F., *Fractal block coding of images,* Electronic Letters 28, 1 (1992) 1053-1055.

[MoNi94] Monro, D. M., Nicholls, J. A., *Real time fractal video for personal communications,* Fractals 2, 3 (1994) 391-394.

[MoWo94] Monro, D. M., Woolley, S. J., *Fractal image compression without searching,* Proc. ICASSP 1994.

[NeGa] Nettleton, D. J., Garigliano, R., *Evolutionary algorithms and a fractal inverse problem,* Department of Computer Science, University of Durham Science Site, Durham, UK.

[Nels91] Nelson, M., *The Data Compression Book,* M&T Books, Redwood City, CA, 1991.

[Nova93a] Novak, M., *Attractor coding of images,* Thesis No. 382, Information Theory Group, Linkoping University, 1993.

[Nova93b] Novak, M., *Attractor coding of images,* Picture Coding Symposium, Lausanne, 1993.

[ØBLMK94] Øien, G. E., Baharav, Z., Lepsøy, S., Malah, D., Karnin, E., *A new improved collage theorem with applications to multiresolution fractal image coding,* to appear in Proc. ICASSP, 1994.

[PeHo91] Pentland, A., Horowitz, B., *A practical approach to fractal-based image compression,* SPIE Visual Communications and Image Processing (1991) 467-474.

[PeJüSa93] Peitgen, H.-O., Jürgens, H., Saupe, D., *Chaos and Fractals: New Frontiers of Science*, Springer-Verlag, New York, 1993.

[Penn93] Pennebaker, W. B., Mitchell, J. L., *JPEG Still Image Data Compression Standard,* Van Nostrand Reinhold, New York, 1993.

[PTVF92] Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., *Numerical Recipes in C,* Cambridge University Press, Cambridge, 1992.

[Ross70] Ross, G. J. S., *The efficient use of function minimization in nonlinear maximum-likelyhood estimation,* Applied Statistics, 19 (1970) 205-221.

[Ross90] Ross, G. J. S., *Nonlinear Estimation*, Springer-Verlag, New York, 1990.

[Saup94a] Saupe, D., *Breaking the time complexity of fractal image compression,* Technical Report, Institut für Informatik, Universität Freiburg, 1994.

[Saup94b] Saupe, D., *Visualizing fractal image compression,* in: *Proc. Eurographics Workshop on Visualization in Scientific Computing,* Rostock, 1994.

[Saup94c] Saupe, D., *From classification to multi-dimensional keys,* in: *Fractal Encoding -- Theory and Applications to Digital Images,* Y. Fisher (ed.) Springer-Verlag, New York, 1994.

[SaHa94] Saupe, D., Hamzaoui, R., *A guided tour of the fractal image compression literature,* Report, Universität Freibürg, May 1994.

[SciAm88] *Fractal shorthand,* Scientific American 258, 2 (1988) 28.

[Stor88] Storer, J. A., *Data Compression: methods and theory,* Computer Science Press, Maryland, 1988.

[Stor92] Store, J. A., *Image and Text Compression,* Kluwer Academic Publishers, Massachusetts, 1992.

[VeRu94] Vences, L., Rudomin, I., *Fractal compression of single images and image sequences using genetic algorithms,* Report, Computer Science Department, Instituto Tecnologico de Estudios Superiores de Monterrey, Campus Estado de Mexico, 1994.

[ViHa93a] Vines, G., Hayes, M. H., *Adaptive IFS image coding with proximity maps,* Proc. ICASSP 5 (1993) 349-352.

[ViHa93b] Vines, G., Hayes, M. H., *Nonlinear address maps in a one-dimensional fractal model,* IEEE Trans. on Signal Processing 41, 4 (1993) 1721-1724.

[Vine93] Vines, G., *Signal Modeling with Iterated Function Systems,* Ph.D. Thesis, Georgia Institute of Technology, May 1993.

[WaKa86] Wallach, E., Karnin, E., *A fractal based approach to image compression,* Proc. ICASSP (1986) 529-532.

[Wall91] Wallace, G. K., *The JPEG Still Picture Compression Standard,* Communications of the ACM, 34, 4 (1991) 30-44.

[Will71] Williams, R. F., *Compositions of contractions,* Bol. Soc. Brasil. Mat. 2 (1971) 55-59.

[With89] Withers, D. W., *Newton's method for fractal approximation,* Constructive Approximation, Vol. 5 (1989) 151-170.

[Zats95] Zatsman, A., *A short introduction to Vector Quantization*, in: *FAQ of comp.compression*, ftp://rtfm.mit.edu:/pub/usenet/news.answers/compression-faq/part2, 1995

APPENDIXES

APPENDIX A

DECODING SEQUENCES OF CLOWN USING PIFS AND CENTERED PIFS

Fisher's PIFS

1st iteration, RMS = 58.14

2nd iteration, RMS = 32.88

3rd iteration, RMS = 19.58

Centered PIFS

1st iteration, RMS = 17.61

2nd iteration, RMS = 11.45

3rd iteration, RMS = 7.97

DECODING SEQUENCES OF CLOWN USING PIFS AND CENTERED PIFS

Fisher's PIFS (continued)

4th iteration, RMS = 10.03

5th iteration, RMS = 8.04

6th iteration, RMS = 7.76

Centered PIFS (continued)

4th iteration, RMS = 7.73

5th iteration, RMS = 7.73

6th iteration, RMS = 7.72

DECODING SEQUENCES OF CLOWN USING PIFS AND CENTERED PIFS

Fisher's PIFS (continued)

7th iteration, RMS = 7.73

8th iteration, RMS = 7.73

9th iteration, RMS = 7.73

Centered PIFS (continued)

7th iteration, RMS = 7.73

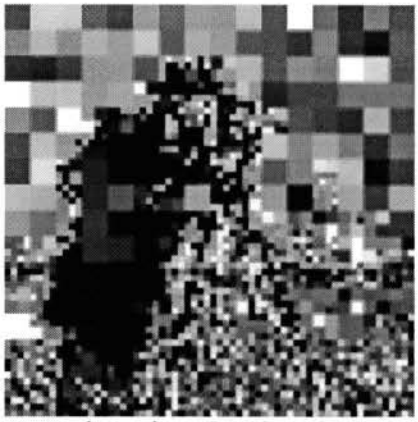
8th iteration, RMS = 7.73

9th iteration, RMS = 7.73

APPENDIX B

DECODING SEQUENCES OF CAMERA USING PIFS AND CENTERED PIFS

Fisher's PIFS



1st iteration, RMS = 60.91

2nd iteration, RMS = 33.69

3rd iteration, RMS = 21.02

Centered PIFS

1st iteration, RMS = 19.71

2nd iteration, RMS = 14.32

3rd iteration, RMS = 11.14

DECODING SEQUENCES OF CAMERA USING PIFS AND CENTERED PIFS

Fisher's PIFS (continued)

4th iteration, RMS = 13.52

5th iteration, RMS = 11.61

6th iteration, RMS = 11.19

Centered PIFS (continued)

4th iteration, RMS = 11.12

5th iteration, RMS = 11.11

6th iteration, RMS = 11.12

Pg. # 120 Not missing

Author has Numbered Pgs.

DECODING SEQUENCES OF CAMERA USING PIFS AND CENTERED PIFS

Fisher's PIFS (continued)

7th iteration, RMS = 11.13

8th iteration, RMS = 11.12

9th iteration, RMS = 11.11

Centered PIFS (continued)

7th iteration, RMS = 11.12

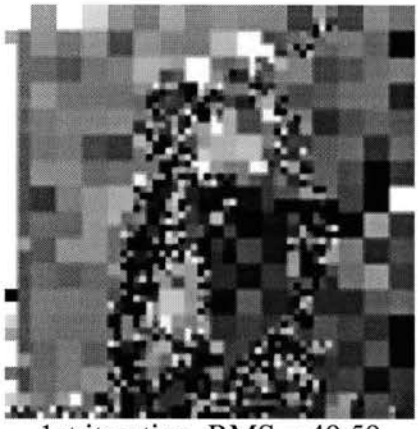
8th iteration, RMS = 11.12

9th iteration, RMS = 11.12

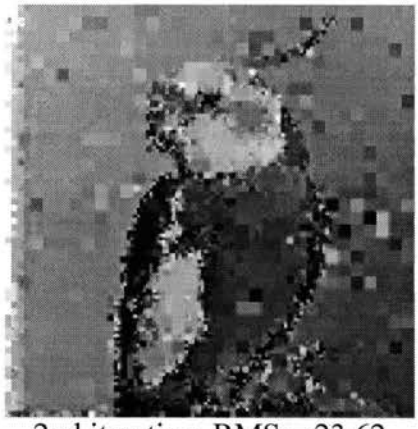
APPENDIX C

DECODING SEQUENCES OF HAMLET USING PIFS AND CENTERED PIFS

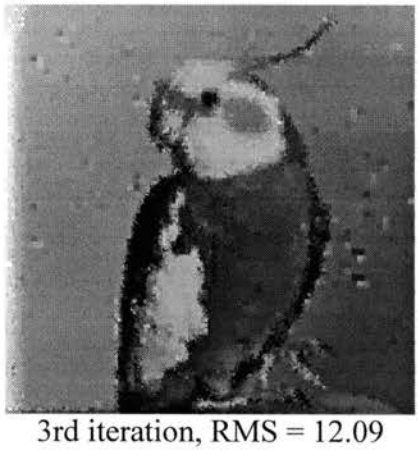
Fisher's PIFS



1st iteration, RMS = 49.59

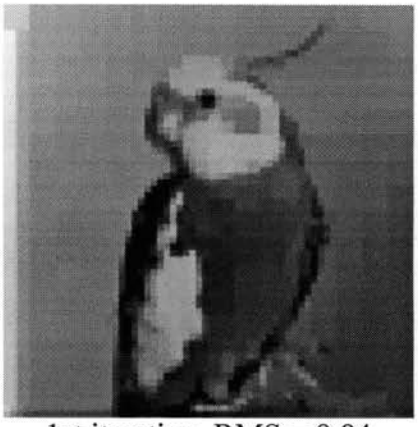


2nd iteration, RMS = 23.62

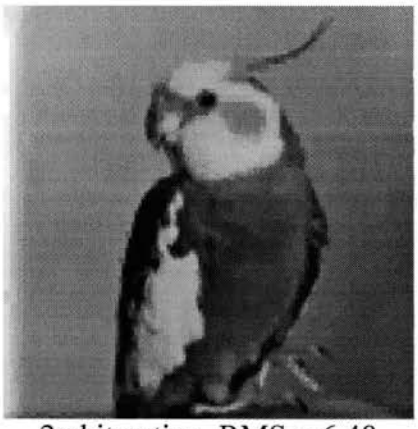


3rd iteration, RMS = 12.09

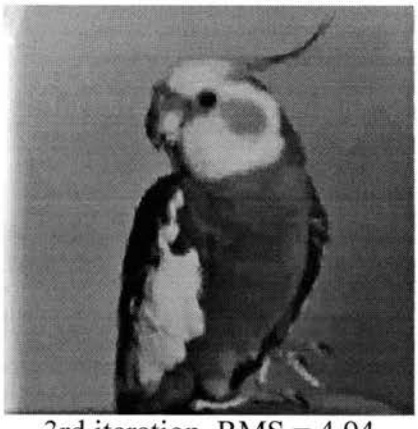
Centered PIFS



1st iteration, RMS = 9.84



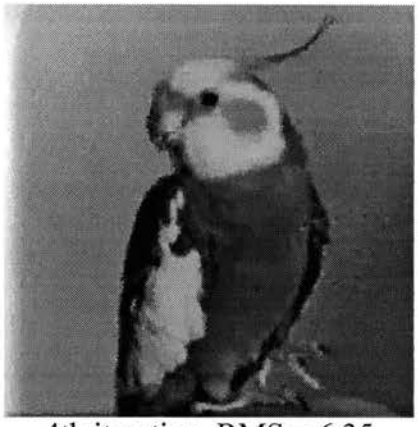
2nd iteration, RMS = 6.48



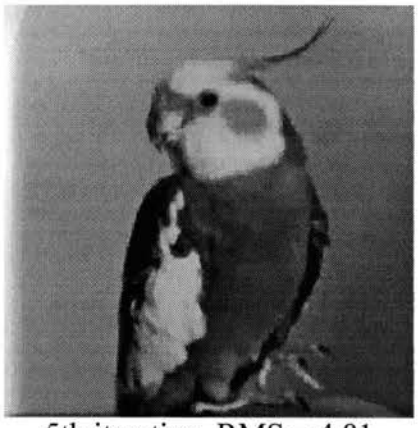
3rd iteration, RMS = 4.94

DECODING SEQUENCES OF HAMLET USING PIFS AND CENTERED PIFS

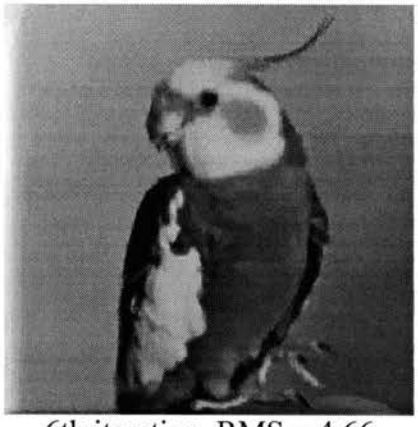
Fisher's PIFS (continued)



4th iteration, RMS = 6.35

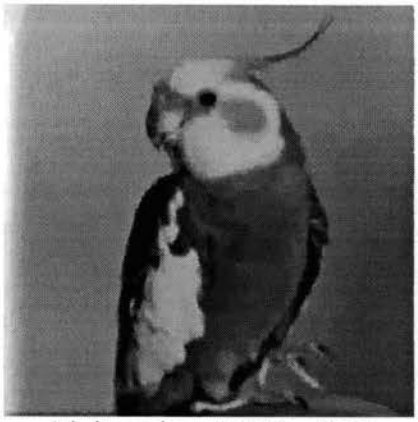


5th iteration, RMS = 4.81

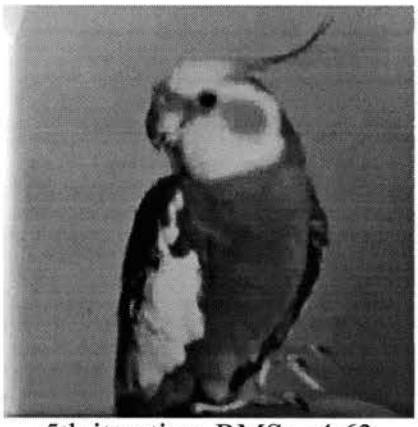


6th iteration, RMS = 4.66

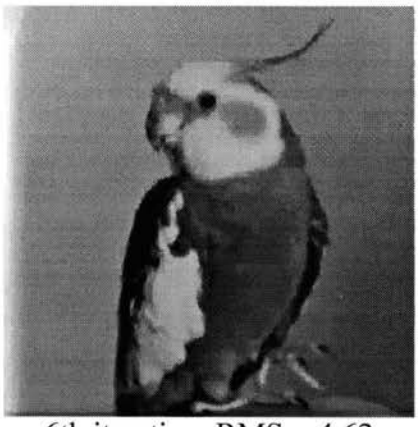
Centered PIFS (continued)



4th iteration, RMS = 4.64



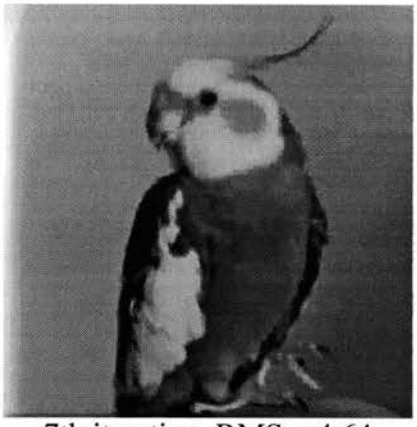
5th iteration, RMS = 4.63



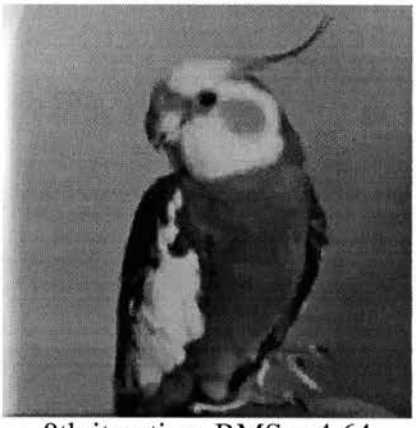
6th iteration, RMS = 4.62

DECODING SEQUENCES OF HAMLET USING PIFS AND CENTERED PIFS

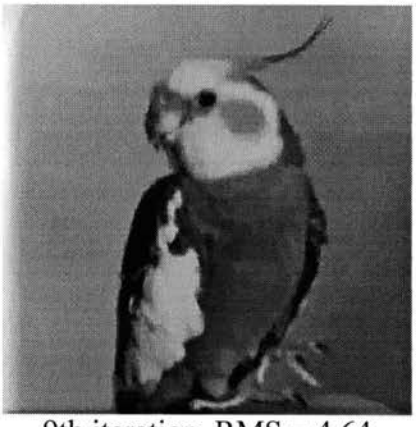
Fisher's PIFS (continued)



7th iteration, RMS = 4.64

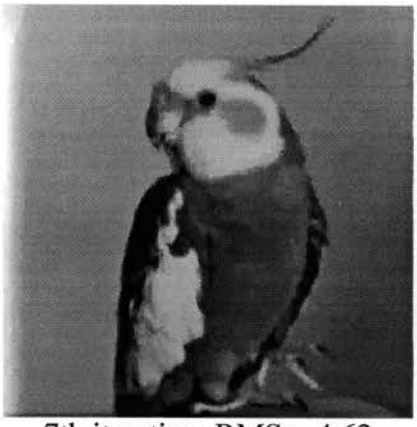


8th iteration, RMS = 4.64

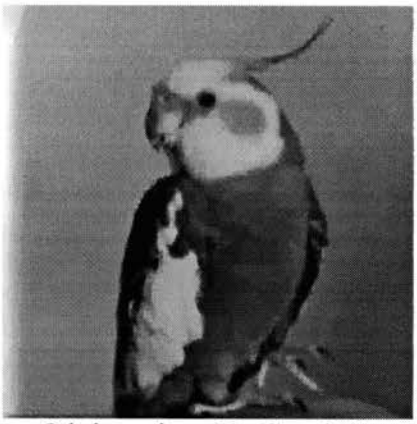


9th iteration, RMS = 4.64

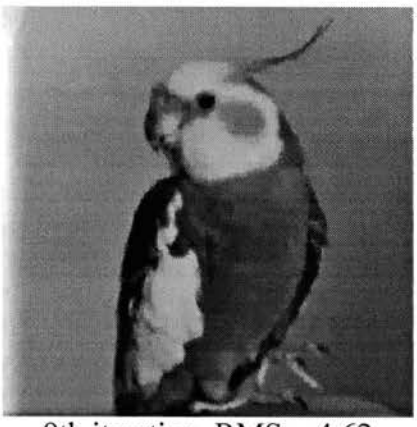
Centered PIFS (continued)



7th iteration, RMS = 4.62



8th iteration, RMS = 4.62

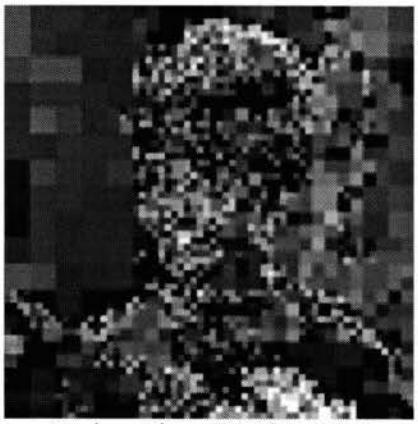


9th iteration, RMS = 4.62

APPENDIX D

DECODING SEQUENCES OF KGIRL USING PIFS AND CENTERED PIFS

Fisher's PIFS



1st iteration, RMS = 39.6

2nd iteration, RMS = 22.62

3rd iteration, RMS = 13.99

Centered PIFS

1st iteration, RMS = 12.17

2nd iteration, RMS = 8.37

3rd iteration, RMS = 6.80

DECODING SEQUENCES OF KGIRL USING PIFS AND CENTERED PIFS

Fishers's PIFS (continued)

4th iteration, RMS = 8.30

5th iteration, RMS = 6.90

6th iteration, RMS = 6.70

Centered PIFS (continued)

4th iteration, RMS = 6.66

5th iteration, RMS = 6.66

6th iteration, RMS = 6.66

DECODING SEQUENCES OF KGIRL USING PIFS AND CENTERED PIFS

Fisher's PIFS (continued)

7th iteartion, RMS = 6.69

8th iteration, RMS = 6.69

9th iteration, RMS = 6.69

Centered PIFS (continued)

7th iteration, RMS = 6.66

8th iteration, RMS = 6.66

9th iteration, RMS = 6.66

VITA

Tung-Ming Koo

Candidate for the Degree of

Doctor of Philosophy

Thesis: IMPROVED FRACTAL IMAGE COMPRESSION: CENTERED BFT WITH QUADTREES

Major Field: Computer Science

Biographical:

Personal Data: Born in Taiwan, R.O.C., on February 1, 1962, the son of Tseng-Lu Ku and Hsiu-Yuin Pan.

Education: Graduated from Ming-Dao Senior High School, Taichung, Taiwan in July 1979; received the Bachelor of Science degree in Civil Engineering from National Chung-Hsing University, Taichung, Taiwan in July 1983; received Mater of Science degree in Computer Science from University of Massachusetts, Lowell, in December 1988; completed the requirements for the Doctor of Philosophy degree at Oklahoma State University in December, 1995.

Professional Experience: Programmer, Gain Computer Company, Taiwan, February 1986 to July 1986. Faculty Assistant, University of Massachusetts, Lowell, August 1987 to August 1988. Assistant Engineer, Computer and Communication Research Laboratories, Industrial Technology Research Institute (ITRI), Taiwan, March 1989 to July 1991. Lecturer, Chung-Yuan Christian University, Taiwan, August 1989 to June 1991. Laboratory Assistant, Department of Agriculture Engineering, Oklahoma State University, January 1992 to August 1992. Teaching Assistant, Department of Computer Science, Oklahoma State University, August 1992 to May 1995.