

A FAST EDGE PRESERVING FRACTAL SYSTEM

By

NIKKI McCLATCHEY BRUNER

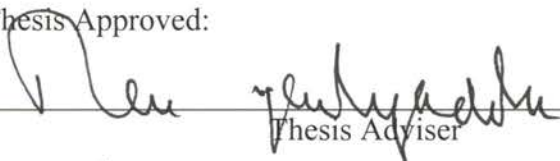
Bachelor of Science
Oklahoma State University
Stillwater, Oklahoma
1987

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1991

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
May, 1998

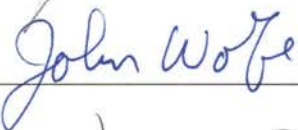
A FAST EDGE PRESERVING FRACTAL SYSTEM

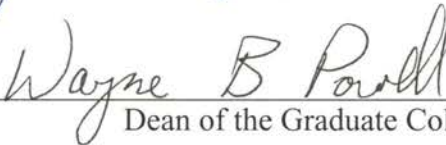
Thesis Approved:


Thesis Adviser






John Wolfe


Wayne B Powell
Dean of the Graduate College

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude and appreciation to Dr. Rao Yarlagadda, my thesis advisor, for his guidance, support, and friendship during the entire period of my dissertation research. His mentoring paved the way for the successful completion of this thesis. In addition, I would like to express my appreciation to my other committee members, Dr. Keith Teague, Dr. Scott Acton, and Dr. John Wolfe for their help and support.

I want to acknowledge that this work was supported by U.S. Army Research Office, contract #DAAH04-95-1-0463, Sandia National Laboratories, contract #AE-9595, and the Southern Regional Education Board Minority Doctoral Scholars Programs.

Finally, I want to give my thanks to my husband, Martin Bruner, for his invaluable support of my work. For all my success, I am grateful to my parents, George and Saroeun McClatchey, for their love and help. Finally, I would like to recognize my daughters, Jordan and Sierra, who provided the primary motivation for this endeavor.

TABLE OF CONTENTS

Chapter	Page
1 INTRODUCTION.....	1
1.1 CLASSIFICATION OF FRACTAL IMAGE COMPRESSION.....	4
1.2 QUANTITATIVE MEASURES.....	6
1.3 OVERVIEW OF RESEARCH	8
2 LITERARY SURVEY	10
2.1 FRACTAL IMAGE COMPRESSION	10
2.1.1 <i>Algorithms</i>	10
2.1.2 <i>Partitioning</i>	13
2.1.3 <i>Computational Complexity</i>	14
2.1.4 <i>Mathematical Support</i>	15
2.2 JACQUIN’S FRACTAL BLOCK CODING	16
2.3 R-TREES DATA STRUCTURES	19
2.4 DIFFUSION TECHNIQUES	21
2.5 SUMMARY OF LITERARY SURVEY	23
3 FRACTAL BLOCK CODERS	24
3.1 MATHEMATICAL BASIS FOR ITERATED FUNCTION SYSTEMS	24
3.2 FIXED BLOCK FRACTAL CODER.....	30
3.3 ERROR DUE TO INACCURATE MAPPINGS.....	35
3.4 METRIC BASED QUAD-TREE FRACTAL CODER	42
3.5 FAST FRACTAL IMAGE COMPRESSION USING R-TREES.	46

3.6	SUMMARY OF CURRENT METHODS.....	49
4	A FAST EDGE PRESERVING FRACTAL SYSTEM.....	51
4.1	ITERATIVE ERROR COMPENSATION TECHNIQUE	52
4.2	WAVELET R-TREE SEARCH FOR FRACTAL CODERS	58
4.3	EDGE BASED QUAD-TREE PARTITIONING METHOD	61
4.4	EDGE PRESERVING DIFFUSION TECHNIQUES.....	69
4.4.1	<i>Mathematical Development</i>	69
4.4.2	<i>Using the Original Image as a Scalar Function in Diffusion Equation</i>	71
4.4.3	<i>Edge Based Scalar Function</i>	75
5	CONCLUSIONS	81
5.1	CONTRIBUTIONS OF FEPFS	82
5.2	FUTURE AREAS OF RESEARCH	84
	REFERENCES.....	86

LIST OF TABLES

	Page
Table 1.1: Number of bits for fractal code paramters.	7
Table 2.1: Transformations for fractal block coding.....	18
Table 3.1: Eight symmetries in local contraction transformations.	29
Table 3.2: Partial listing of fractal code.	31
Table 3.3: Results for using 8x8 and 16x16 range sizes.	34
Table 4.1: Comparison of quantitative measures for diffusion.....	78
Table 4.2: Quantitative measures for splitting level without diffusion.....	79
Table 5.1: Comparison of results for 512 x 512 Lena.....	82

LIST OF FIGURES

	Page
Figure 1.1: Example of a fractal image.	2
Figure 2.1: Collection of data objects into r-tree structure.	20
Figure 2.2: Spatial representation of data objects.	20
Figure 3.1: Iterations of the fractal code.	33
Figure 3.2: Attractor of fractor code for different range block size.	34
Figure 3.3: Comparision of 'Lena' (128x128) with the attractor.	36
Figure 3.4: Block PSNR versus Error Metric.	37
Figure 3.5: Classification of the original image into areas of PSNR values.	39
Figure 3.6: Representations of the error metric and PSNR for image blocks.	40
Figure 3.7: Residual error image.	41
Figure 3.8: Comparison of the attractor at different error tolerance levels.	44
Figure 3.9: PSNR versus error tolerance for metric based quad-tree coder.	45
Figure 3.10: Compression versus error tolerance for a metric based quad-tree coder.	45
Figure 3.11: PSNR versus compression for a metric based quad-tree coder.	46
Figure 3.12: Encoding time for different fractal coders.[40]	47
Figure 4.1: Results for different block sizes in fractal block coding.	53
Figure 4.2: Flowchart of the iterative error compensation technique.	55
Figure 4.3: Results of the iterative error compensation technique.	57
Figure 4.4: PSNR versus dimensions.	60
Figure 4.5: Search time versus dimensions.	60
Figure 4.6: Edge images for 'Lena'	63

Figure 4.7: Comparison attractor at different edge splitting level.	64
Figure 4.8: PSNR versus edge level in edge based partitioning.	65
Figure 4.9: Compression versus edge level in edge based partitioning.	66
Figure 4.10: PSNR versus compression for edge based partitioning.	67
Figure 4.11: Attractors of different partitioning methods.	68
Figure 4.12: Laplacian and gradient images of Lena.	72
Figure 4.13: Residual image for diffusion.	73
Figure 4.14: Comparison of attractor diffused at different levels.	74
Figure 4.15: Comparison of the attractor (a) and the diffused image (b).	77

LIST OF IMPORTANT ACRONYMS

FEPFS	Fast Edge Preserving Fractal System
FFIC	Fast Fractal Image Compression
IFS	Iterated function system
PSNR	Peak signal to noise ratio

CHAPTER I

1 INTRODUCTION

The advances in computer technology in the last decade have redefined computer information from mainly text to a multimedia format of text, sounds, and images. Multimedia applications are no longer limited to 'high-ended' workstations, but are currently being used on 'entry-level' personal computers in the home. Data compression techniques have played a part in this historic event by providing methods for sounds and images to be stored compactly and processed efficiently. The amount of data generated to represent sounds or images can be significant affecting storage requirements and processing time.

Data compression addresses the problem of reducing the amount of data it requires to represent information like text, sounds, and images. Image compression deals with the reduction of data required to represent digital images. Generally, there are two types of compression, lossless and lossy. In lossless compression, the data can be recovered without losing any information. In the lossy case, information like images are represented in a manner such that the reconstructed images are acceptable to the end user. This allows for greater reduction in the information; although, the images may be degraded. This research focuses on the potential of fractal image compression, a lossy technique, for compressing digital images for 'real time' applications.

Almost everyone has seen the exciting computer generated images of mountain ranges, interplanetary scenes, exotic plants, and alien landscapes. Figure 1.1 illustrates a fractal image example. A computer program using very little base information to start

with constructs these seemingly real images. Not surprisingly, the creation of these images is based upon a class of geometry known as fractals.

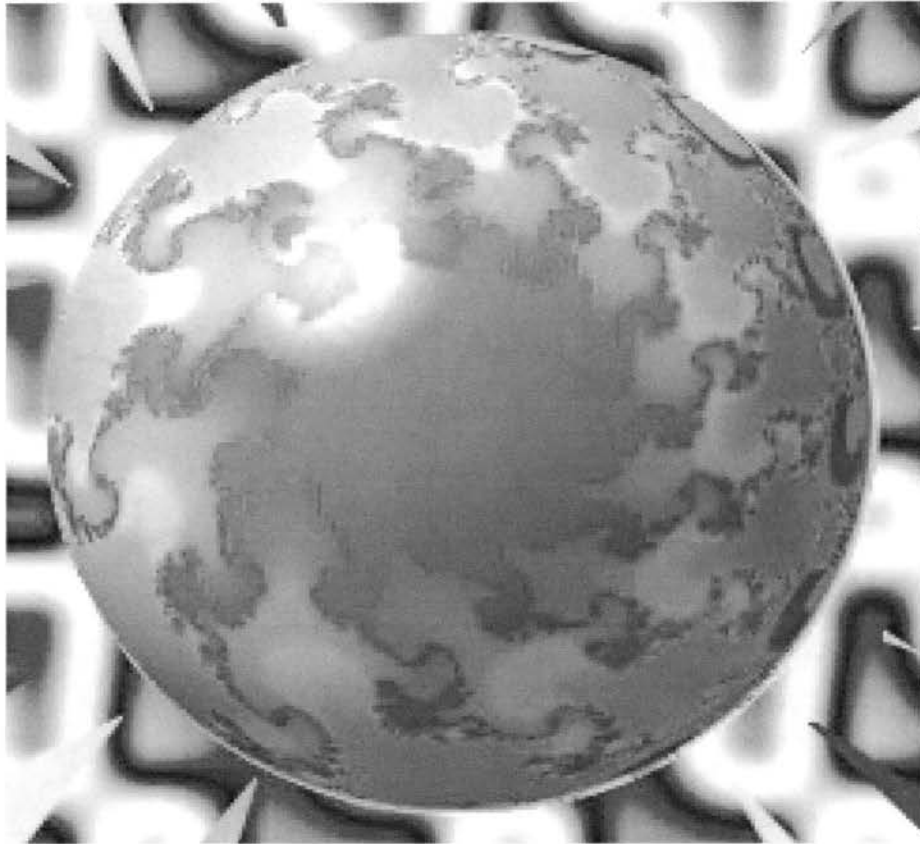


Figure 1.1: Example of a fractal image.

Fractals are fixed points to certain equation sets [3]. The self-similar property of fractals allowed it to be used to describe natural features like clouds, leaves on a tree and other natural phenomenon. Clouds are considered self-similar since parts of a cloud compared with the whole are very much the same. Although digital images as a whole are not self-similar, there are regions of self-similarity [66].

In a digital image, numerous smaller regions look like larger regions in the image. Fractal image compression relates these areas of local self-similarity at different scale

[49]. Taking advantage of this correlation, the fractal code pairs up similar regions creating a list of 'optimal' pairs. Due to the sizes of digital images, searching for 'optimal' pairs of similar blocks in the image constitutes the major computation load for fractal coders. The prohibitive CPU time needed for searching for the 'optimal' pairs is a major obstacle to the use of fractal image compression for 'real time' applications.

Applying this list of fractal transform parameters approximates the original image. The reconstructed image is the attractor of the fractal code. The accuracy of the attractor depends on the degree of similarity between the larger to smaller mapping. In order to increase the degree of similarity, the first automatic fractal coder proposed by Jacquin [35] uses quad-tree splitting to partition the image into two block sizes. The quad-tree splitting method divides a large square or block into four equal sized smaller squares. In smooth areas of the image, using the larger block size gives higher compression ratios. In areas of edges, the smaller block size increases the accuracy of the fractal mapping.

Degradation of edges and loss of edge information occur in the attractor due to the limitation of fractal code to describe the similarities in the image with contractive affine transformations. The limitation of accurately describing the image at large block sizes reduces the effective compression ratios of fractal coders.

In order to create a fast edge preserving fractal system for compression of digital images in the field, our research deals with overcoming these limitations. We will investigate ways of decreasing the encoding times and increasing the preservation of edge information. First, we will classify fractal image compression, provide the quantitative measure used in our research, and give an overview of our research.

1.1 Classification of Fractal Image Compression

This section presents a simple classification of fractal image compression relative to current compression methods [40]. As mentioned earlier, image compression methods are categorized into two major areas: lossless and lossy methods. In lossless methods, an exact duplicate of the original image can be reconstructed from the compressed data. In lossy methods, the attractor of the fractal code approximates the original, thus allowing for higher compression ratios. Compression ratio measures the number of bits representing the coded image with respect to the original image. A high compression ratio means it takes fewer bits to represent the digital image in the coded format than in the original format. Although there is loss of fidelity due to artifacts and distortion, this loss is minimized such that the attractor of the fractal code still ‘looks good’ to the human eye.

In lossy methods, there are two sub-categories of interest: transform and vector quantization techniques. Transform techniques constitute the broadest area of lossy methods. Although transform techniques can be lossless, the methods generally achieve compression by eliminating certain coefficients in the transform. This area includes the Discrete Fourier Transform (DFT), the Discrete Cosine Transform (DCT), wavelets, Karhunen-Loeve Transform (KLT), and others. The popular JPEG, Joint Photographic Experts Group, standard for sequential compression is based on the DCT method [5]. The image is divided into smaller blocks (generally 8x8 pixels). Each block is transformed using the FDCT (forward Discrete Cosine Transform) resulting in a set of quantized coefficients. Coefficients below a certain ‘user-defined’ level are discarded. The remaining set is entropy-encoded using arithmetic or Huffman coding. The entropy-encoded DCT coefficients represent the image in a format that requires less memory storage than the original image. In a performance study by Fisher, Rogovin, and Shen

[23], fractal image compression proves itself viable in comparison with transform methods by giving comparable compression ratios and fidelity results.

Wavelet image compression consists of the image data decorrelating transform and the data symbol entropy [46]. The wavelet decompositions are obtained using two approaches: multiresolution analysis and scaling functions [48]. In general, wavelet compression algorithms are based on decorrelating the image using subband decomposition and biorthogonal wavelet functions. The significant coefficients approximating the image are entropy coded. In comparison to wavelet compression, fractal coding techniques are comparable for compressing (512x512) images [21]. Wavelet techniques perform better in terms of PSNR at lower compression ratios (less than 5) however, the wavelet and fractal techniques perform roughly the same for higher compression ratios.

Like the transform techniques, vector quantization (VQ) techniques also partition the image into an array of blocks or smaller sections. In the standard VQ methods, these image blocks are compared to a predefined indexed set of image blocks. Each image block is coded by the index number of the block that most closely approximates it based on some criteria such as least squares. Since the image is reconstructed from this set of blocks, this set must be transmitted or installed at the decoding site.

Although fractal image compression has been called 'self-VQ' [22], it differs from the standard VQ techniques by the contractive nature of the fractal mapping and the necessity of the predefined set of image blocks. In order to achieve high compression ratios, fractal coders use information contained in large blocks or regions of the image to fill in small-scale details. By copying and shrinking these large blocks, these coders approximate smaller scaled blocks in the image, creating a list of fractal mappings. Applying these mappings on any initial image approximates the original image. Since these mappings only approximate the original image, fractal image compression is a lossy method.

1.2 Quantitative Measures

Three quantitative measures, compression ratio, peak signal to noise ratio (PSNR), and a measure of edge preservation (ξ), provide a quantitative evaluation of our research. We use these measures to compare our methods with other techniques and to gauge the performance of our research. The amount of compression gained and the closeness of the attractor of the fractal code, v , to the original image, u , dictate the performance.

For fractal image compression, compression ratio is

$$\frac{\text{\# of bits in the original image}}{\text{\# of bits in the fractal code}} : 1 . \quad (1.1)$$

According to (1.1), the number of bits in the fractal code determines the compression ratio. For our research, Table 1.1 gives approximately the number of bits required to represent the different parameters used in the fractal code for coding images [20]. Since the fractal code is a list of parameters, entropy coding the fractal code can produce higher compression ratio.

An example of entropy coding for image coders is run-length/Huffman coding. Run length encoding [58] compresses runs of identical symbols in the data. It encodes the run of symbols as a symbol and a count. Huffman coding [58] encodes the symbols in the data in proportion to its information content. Fewer bits are used to code symbols with higher probability of occurrence while more bits are used to code symbols with lower probability of occurrence in the data set. These are lossless techniques used to compress information like text. Entropy coding the fractal parameters gives further compression.

Peak signal-to-noise ratio, PSNR, is a measure of fidelity commonly used in image compression. PSNR gives general indication of the quality of the attractor of the fractal code in respect to the original image. PSNR [19] is

$$\text{PSNR} = 20 \log_{10} \left(\frac{b}{rms} \right) \quad (1.2)$$

where b is the largest possible value of the signal and rms is the usual root mean square difference between the original image and the attractor of the fractal code.

Table 1.1: Number of bits for fractal code paramters.

Coefficients	Number of Bits
Scaling Term	5
Offset Term	7
Domain Position	14
Symmetry	3
Quad-tree	1

We use a measure of edge preservation developed by Sattar, Floreby, Salomonsson, and Lovstrom [68] to measure the correlation of edge information between the original image and the attractor of the fractal code. The measure of edge preservation (ξ) is

$$\xi = \frac{\sum_{x,y} \Delta v' \Delta u'}{\sum_{x,y} \Delta v' \Delta v' \sum_{x,y} \Delta u' \Delta u'} \quad (1.3)$$

where $\Delta u'$ and $\Delta v'$ are sharpened (high pass filtered) versions of the original image and

attractor of the fractal code normalized by their means. The measure of edge preservation approaches unity when the approximation is close to the original image.

1.3 Overview of Research

The need for a fast compression image system with high compression ratio for ‘real time’ applications like surveillance is the primary motivation for this research. Due to its strength of high compression ratios, fractal image compression is a viable choice. Although these applications require less fidelity, the attractor of the fractal code must provide enough detail to identify objects of interest in the image. A significant amount of information needed for identification is in the ‘edges’ of the image. The need for high compression ratio dictates the use of large block sizes in fractal compression. Inaccurate fractal mappings at large block sizes increase losses of edge information, discontinuities at boundaries and blocking effects. This research focuses on finding a method of preserving edges to a reasonable extent in the attractor of the fractal codes while maintaining high compression ratios with a fast fractal compression scheme.

A fast edge preserving fractal system, FEPFS, is produced by our research. Since searching for ‘optimal’ pairs of similar blocks in the image constitutes the major computation load for fractal coding, FEPFS uses a wavelet r-tree search engine to reduce the searching time to seconds. Incorporation of an edge based quad-tree partitioning allows the image to be partitioned along edges to help preserve edges. This partitioning scheme also provides a front-end loader for the search engine. Since partitioning does not insure the retention of significant edges, FEPFS uses diffusion techniques to preserve significant edge information. By expanding the basic diffusion equation to contain a scalar function based on the edges of the original image, FEPFS directs the diffusion process to smooth along the direction of significant edges and sharpen in the direction of

the edges. In this manner, FEPFS restores edge information and reduces discontinuities and blocking effects.

This dissertation assesses the contributions of FEPFS by providing a literary survey of fractal image compression research, reviewing current implementations in fractal block coding, and detailing the development and performance of an edge based quad-tree partitioning scheme, a wavelet based r-tree search engine, and diffusion techniques in FEPFS. In conclusion, the contributions of our research along with future research directions in the area of fractal image compression are summarized.

CHAPTER II

2 LITERARY SURVEY

This chapter provides a survey of the research in fractal image compression, r-trees, and diffusion techniques. This survey deals mainly with fractal image compression although it briefly deals with r-trees and diffusion techniques.

2.1 Fractal Image Compression

In this section, we categorize the current literature on fractal image compression into four major areas of research, algorithms, partitioning, computation complexity, and mathematical support. Although many of the papers overlap several areas, the papers are categorized according to their major contribution.

2.1.1 Algorithms

The idea of using fractal techniques to exploit the local regions of self-similarity in digital images was introduced by Michael Barnsley in 1988 [6]. Barnsley modeled digital images as mathematical spaces and used fractal techniques to represent images as sets of equations. Barnsley introduced fractal compression as a better way to compress digitized images, promising compression ratios in the range of 10,000. He achieved this compression ratio only for very specialized images and not for real life digital images. Barnsley [5] later developed the theory of iterated function systems (IFS) as an extension

of classical geometry. Using affine transformations (combinations of rotation, scaling, and translation of the coordinate axes in a metric space) to relate parts of the image, digital images were coded by these relationships forming the basis for fractal block coding.

A student of Barnsley, Arnaud Jacquin [35],[36],[37] simplified this complicated process for gray scale images. In 1990, Jacquin [35] presented the first fully automated practical implementation of encoding gray scale images called fractal block coding. His fully automated algorithm for fractal image compression provided the substructure for many current fractal compression techniques including FEPFS.

Results [37] of the fractal block coding observed by Jacquin showed overall good fidelity between the decoded image and the original. He attributed blocky artifacts in the attractor of the fractal code to the use of square range blocks. The quality of the decoded image depended heavily on block classification and analysis. Large blocks, blocks with weak edges and blocks with strongly contrasted textures were more prone to distortion or error in the attractor of the fractal code.

Oien, Lepsoy, and Ramstad [63] introduced an optimization procedure for finding the optimal transform within a class of affine transformations using the inner product space theory. Using the projection theorem, they determined the optimal coefficients for their optimal transform by solving a set of orthogonality equations. These equations were created from the inner product of the error measure and the basis sets. Oien, Lepsoy, and Ramstad [64] reduced the complexity of their fractal coder by combining their orthogonalization procedure with an automatic block classification procedure. Oien [61] detailed the quantization of parameters for their block based fractal coders.

Gharavi-Alkhansari and Huang [26] developed an orthonormal basis approach for fractal block coding. Using the Gram-Schmidt procedure, a set of orthonormal basis vectors was created. The fractal code was created from the projection of the image blocks onto this basis.

Monro and Dudbridge [53],[54] introduced a least-square approximation method of fractal coding, known as the Bath fractal transform. Monro [51],[52] coined his method as 'hybrid fractal transform' since it was self-tiling. Monro and Woolley [55],[56],[74] extended and optimized this method. They encoded digital images by tiling each block in the images with a reduced copy of itself using a least-squares criterion. They developed a system of linear equations based on taking the partial derivatives with respect to the fractal parameters and setting them equal to zero. Solving the system of linear equations resulted in the least squares approximation to an image block tiled by itself. This algorithm resulted in faster encoding time, lower compression ratios and less fidelity than fractal block coding.

Incorporation of pyramid methods in fractal coding [1],[8],[10],[13],[17] emphasized the properties of self-similarity at different resolutions. The property of self-similarity at different resolutions is not represented in the general fractal transform used in fractal block coding. Baharav, Malah, and Karin [1] modified the basis fractal transform to include this property. In general, they described the relation between two different fixed points in the attractor, when the attractor is halved or doubled. These interpretations were extended to wavelet representations of fractal compression [14],[72]. Davis [15] generalized fractal block coding as a Haar wavelet subtree quantization scheme. He showed that fractal coders are effective due to their ability to efficiently represent wavelet zerotrees.

Unifying fractal compression algorithm with other image compression algorithms provided a different representation or viewpoint for fractal transforms. Barthel, Schuttemeyer, Voye, and Noll [7] extended the luminance transformation used in fractal image coding to the frequency domain. They reported a reduction of encoding time due to larger block sizes and an improvement in the subjective quality of the image. Lin [38] showed that fractal image coding could be viewed as a generalized form of predictive image coding. Gharavi-Alkhansari and Huang [27] developed a generalized image

block-coding algorithm in which fractal image techniques, block transform and vector quantization methods were special cases of this algorithm.

Decoding a fractal code was generally simple and fast; therefore, most of the research in fractal image compression dealt with finding the fractal transformation describing an image block. Forte [24],[25] referred to the problem of finding the fractal transform to describe an image as the 'inverse problem'. Pei, and Tseng, and Lin [65] developed a method to decode the subimages of a fractal code in parallel with no transient behaviors common to some decoders. Lepsoy, Oien, and Ramstad [44] implemented a fractal coder with a fast non-iterative decoding algorithm. Using 'cut and paste' mappings, the composition of the mappings was applied to the entire image rather than individual blocks. In this method, the coder mimics the decoder sequence. A new map was chosen at every step to keep the approximations close to the original vector.

Use of fractal coding in video applications [9],[22],[43],[59] required the extension of the fractal transform into higher dimensional representation to handle the time component of video frames. Three-dimensional image [57] for solid or volume representation of objects also required extension of the fractal transform to higher dimensions.

2.1.2 Partitioning

In current techniques, fractal coders use partitioning to determine the largest range block possible for a given region in the image based on some fidelity criterion [66]. Partitioning schemes directly affect the compression ratio, coder complexity, and fidelity. Looking at an image, there are regions of detail difficult to cover with a contractive transformation at large block sizes. In these regions, coders use smaller blocks to maintain accuracy in the mappings. In smooth regions of the image, coders use larger block sizes for higher compression ratios. There are three basic methods of

partitioning[66]: quad-tree partitioning, HV (horizontal-vertical) partitioning, and triangular partitioning.

Quad-tree partitioning [71] divides large image blocks into four equally sized sub-blocks in order to meet the fidelity criterion. This process can be repeated on the sub-blocks until the smallest subdivision of the blocks had occurred or the fidelity criterion has been met.

The quad-tree partitioning is a simple partitioning scheme that requires only one bit per block of overhead. A weakness of the quad-tree partitioning method is that it does not try to partition the blocks based on its contents whereas HV partitioning [66] does. In HV partitioning, larger range blocks are recursively partitioned either horizontally or vertically into rectangular blocks. A key point used to determine the size and orientation of the rectangular blocks is the location of the edges.

The adaptive triangle partitioning [16] developed by Davoine, Bertin, and Chassery uses a split and merge approach. This approach guides the evolution and location of triangles. The adaptive triangle partitioning improves the quality of the attractor or decoded image. It reduces the ‘tiling’ or ‘blocky’ artifacts common with the square and rectangular methods.

These methods, HV partitioning and triangle partitioning, are more complicated to implement. They require additional information in the fractal code, thus reducing compression ratios.

2.1.3 Computational Complexity

Fractal coders suffer from high computational complexity due to the extensive library or domain block searches for optimal pairing of blocks in the fractal transform. Computational complexity is directly related to the time it takes for a computer to

compress an image. Some methods to reduce the search time include limiting the search space and using heuristic search techniques.

Limiting the search space includes restricting the domain blocks to nearest neighbors [73], pruning the transform space [69], and using clustering techniques to group the library blocks [11],[64]. One library-clustering algorithm [64] uses a vector quantization algorithm to cluster vectors according to a small set of centroids. Another clustering technique [11] uses a Kohonen neural network to cluster like image blocks. They issued a self-organizing feature map to organize similar features in different resolution of the image. After training the network, the different patterns or features in the image tend to cluster into groups.

The direction of current trends [2],[38],[41],[46] for search techniques has been the incorporation of data tree structures. In the tree structure, similar library vectors are located near each other in parent-children node arrangement based on a set of characteristics. Using the range characteristics, a best match vector or set of vectors can be found for the range vector by searching the nodes of the tree. In Section 2.2, the r-trees data structures are detailed.

2.1.4 Mathematical Support

Bondarendo and Dolnikov [12] expanded on the mathematical concepts and assertions that form the basis for fractal image compression. Oien, Baharav, Lepsoy, Karnin, and Malah [62] extended the collage theorem to multi-resolution fractal image coding. They validated and extended the fundamental mathematical support of fractal image compression.

Contractivity factors [31],[32],[33] provided the mathematical support of the convergence of the fractal transforms to a fixed point. Using spectral theory, the condition for contractivity was satisfied if the spectral radius of the transformation matrix

was less than one. Although, Jacobs, Fisher, and Boss [34] showed that the attractor of the overall IFS system approximates the original image better if some of the transformations in the IFS are allowed to be non-contractive,. They concluded it was not necessary that the individual transformation w_i be contractive. The overall IFS system converged to a fixed point when the mapping W of the overall IFS were eventually contractive. The mapping W was composed of a union of the individual transformation w_i . If the contractive w_i eventually dominated the expansive ones, the mapping W was eventually contractive.

2.2 Jacquin's Fractal Block Coding

Because of its importance to our research, we present Jacquin's fractal block coding [37] in detail. For fractal block coding, data compression ratios ranges from 10:1 to 50:1 depending on the images. This method codes images with a class of discrete image transformations that are contractive with respect to the L_2 metric or measure. Iterating the transformations on any initial image recover an approximation of the original image.

Fractal block coding partitions the image into non-overlapping square range blocks of two different sizes ($B \times B$, $B/2 \times B/2$). The two different sizes allow for regions containing more details to be encoded using the smaller size and for regions containing lesser detail to be coded using the larger size. The smaller size results in less error in the attractor of the fractal code while the larger size yields higher compression ratio.

In order to reduce the coder complexity, fractal block coding classifies domain blocks, D_i , and range blocks, R_i , by their perceptual geometric features. The domain blocks are classified into pools based on edge content in order to reduce the search time. This minimizes the number of comparisons between range blocks and transformed domain blocks to find the 'optimal' pair of domain and range blocks.

Three discrete transformations or mappings describe the three categories of image blocks. These categories include shade blocks, midrange blocks, and edge blocks. Shade blocks are blocks with uniform levels of intensity. Edge blocks contain some degree of edge information. The rest of the blocks consist of midrange blocks.

Shade blocks are coded using just the absorption transformation,

$$\tau_i = g_0 \quad (2.1)$$

where g_0 was the average gray level of the range blocks. Midrange range blocks are coded using spatially contracting transformations. The spatially contracting transformation is given by

$$\tau_i = \alpha_i S_i(D_i) + \Delta g_i \quad (2.2)$$

where α_i is a contrast scaling factor between 0 and 1 and Δg_i is the difference between the average gray level of the R_i and D_i . The geometric part, S_i , represents the spatial contraction of the image blocks from a domain block size to a range block size. These transformations are compositions of contrast scaling, luminance shifts, and spatial reductions. For ‘edge’ range blocks, isometries, shown in Table 2.1, generates geometrically related transformed blocks used to describe details in the image. The transformation takes the form of

$$\tau_i = i_i(\alpha_i S_i(D_i)) + \Delta g_i \quad (2.3)$$

where (2.2) has been expanded to contain i_i , an isometry.

Since the image is coded as a list of transformation parameters, fractal block coding uses entropy coding to compress the list for higher compression ratios. In image compression, quantized parameters or coefficients are generally encoded to increase the compression ratio.

Table 2.1: Transformations for fractal block coding.

Transformation	Equations
Absorption	$(\theta\mu)_{i,j} = g_0$
Luminance shift	$(\tau\mu)_{i,j} = \mu_{i,j} + \Delta g$
Contrast scaling	$(\sigma\mu)_{i,j} = \alpha\mu_{i,j}$
Isometries $\{i_k\}_{0 \leq k \leq 7}$	
identity	$(i_0\mu)_{i,j} = \mu_{i,j}$
orthogonal reflection about mid-vertical axis of block	$(i_1\mu)_{i,j} = \mu_{i,B-1-j}$
orthogonal reflection about mid-horizontal axis of block	$(i_2\mu)_{i,j} = \mu_{B-1-i,j}$
orthogonal reflection about first diagonal ($i = j$) of block	$(i_3\mu)_{i,j} = \mu_{j,i}$
orthogonal reflection about second diagonal ($i + j = B - 1$) of block	$(i_4\mu)_{i,j} = \mu_{B-1-j,B-1-i}$
rotation around center of block by $+90^\circ$	$(i_5\mu)_{i,j} = \mu_{j,B-1-i}$
rotation around center of block by $+180^\circ$	$(i_6\mu)_{i,j} = \mu_{B-1-i,B-1-j}$
rotation around center of block by -90°	$(i_7\mu)_{i,j} = \mu_{B-1-i,j}$

Legend:

μ	pixel gray levels of an image block
g_0	average gray level of an image block
α_i	contrast scaling factor between 0 and 1
Δg	difference between two average gray levels
i_i	isometries
θ	absorption
τ	luminance
σ	contrast

2.3 R-trees Data Structures

Kominek [41] uses r-trees to reduce the search time to seconds for block sizes of 4x4. An r-tree is a dynamic index structure for spatial searching of large databases. Examples of spatial data are lines in 2-dimensional space and volumes in 3-dimensional space[29]. Since spatial data objects cover regions in a multi-dimensional space, the index in r-trees is based on the spatial locations of the data objects [30]. If each record in the database is thought of as a point in a multi-dimensional space, records in database can be retrieved based on their spatial location efficiently and quickly.

Extending this concept to fractal image compression, an image block is a vector in a multi-dimensional space. Searching for the optimal pairing of range and domain blocks consists of finding the record in the library or domain database that best correlates to the range vector.

The data structure for r-trees is a height-balance tree containing leafs of indexed records pointing to the spatial data objects. All the leaf nodes exist at the same level in the tree. The entry of the leaf node consists of the form (X_i, Y_i, I) where X_i and Y_i are the coordinates of the smallest rectangle that bounds the data object and I is the identifier that points to the data object. Non-leaf nodes are of the form (X_i, Y_i, P) where X_i and Y_i are the coordinates of the smallest rectangle that bounds all the rectangles in the child nodes that P points to.

Figure 2.1 show the collection of the data in to a tree structure. In Figure 2.2, the bounding rectangles for the nodes and the data objects are shown in a spatial representation. The tree is searched from the top node or root down to the lowest level. Using the indexed tree structure, only the objects in space of interest are examined. Although more than one of the sub-trees under a node may need to be searched, the tree is maintained in a manner to minimize the search space. A detail description on the implementation of the r-tree structure can be found in [29].

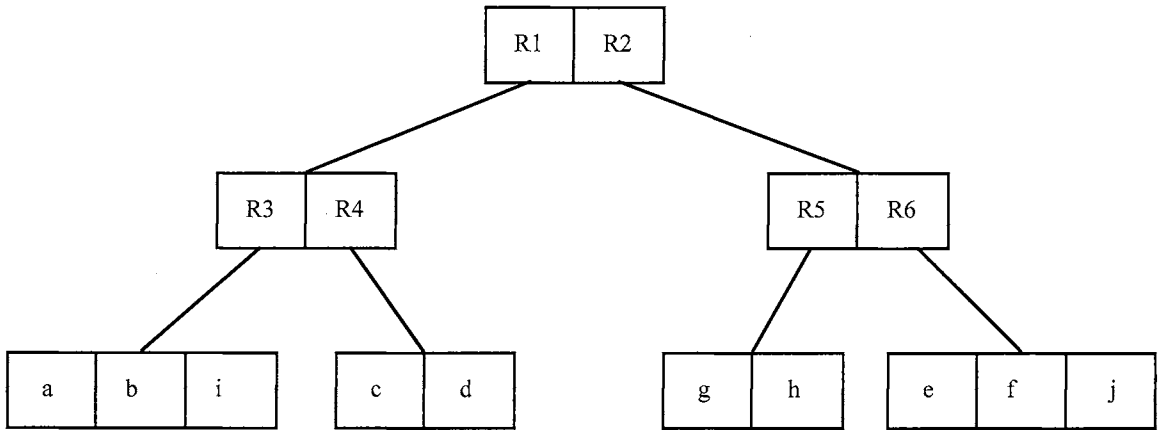


Figure 2.1: Collection of data objects into r-tree structure.

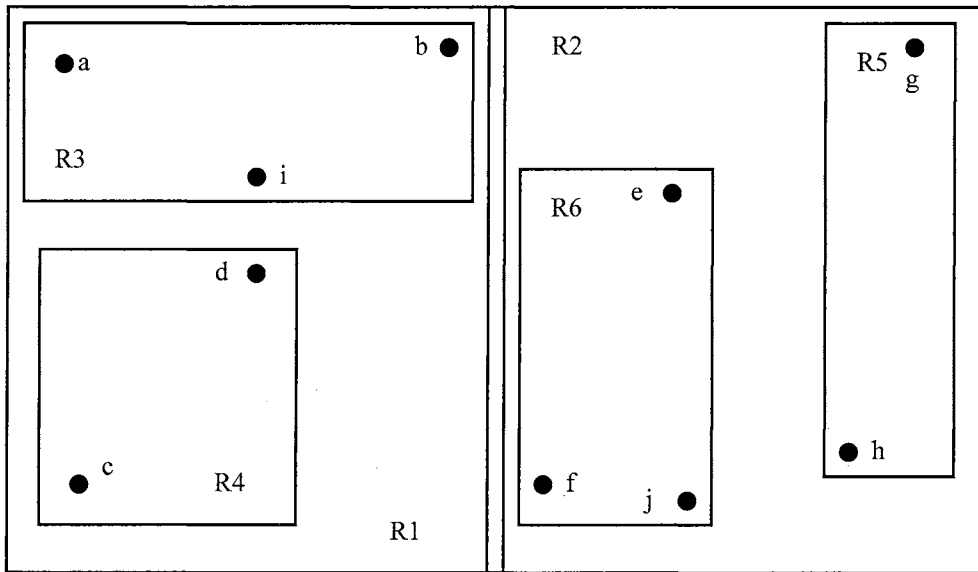


Figure 2.2: Spatial representation of data objects.

2.4 Diffusion Techniques

Current diffusion techniques [18],[42],[70],[76] are used for segmenting images, detecting edges, enhancing images, and restoring images. Choosing a diffusion coefficient as a function of the image gradient results in image enhancement. Enhancement occurs due to smoothing of flat regions to eliminate noise through forward diffusion and sharpening of edge regions through backward diffusion. Forward diffusion is diffusion occurring in the direction orthogonal to the gradient and backward diffusion is in the direction of the local gradient. Unfortunately, current techniques that allow backward diffusion tend to be ill-posed [60][75] causing undesirable results. To overcome this, diffusion techniques based on evolution of curves [18],[42] embedded in higher dimensional surfaces [50] use just forward diffusion to smooth along the direction of the edges to eliminate noise while preserving edges.

This section examines the nonlinear diffusion filtering developed by Perona and Malik [67]. The anisotropic diffusion equation used by Perona and Malik is

$$I_t = \text{div}(c(x, y, t)\nabla I) \quad (2.4)$$

where div is the divergence operator and ∇ is the gradient operator. Choosing the conduction coefficient $c(x, y, t)$ to be

$$c(x, y, t) = g(\|\nabla I(x, y, t)\|) \quad (2.5)$$

will preserve and sharpen the edges for the brightness function $I(x, y, t)$ of the image if $g(\cdot)$ is chosen properly. Discrete implementation of (2.4) by Perona and Malik is given as

$$I_{i,j}^{t+1} = I_{i,j}^t + \lambda [c_N \cdot \nabla_N I + c_S \cdot \nabla_S I + c_E \cdot \nabla_E I + c_W \cdot \nabla_W I]_{i,j}^t \quad (2.6)$$

where

$$\nabla_N I_{i,j} \equiv I_{i-1,j} - I_{i,j} \quad (2.7)$$

$$\nabla_S I_{i,j} \equiv I_{i+1,j} - I_{i,j} \quad (2.8)$$

$$\nabla_E I_{i,j} \equiv I_{i,j+1} - I_{i,j} \quad (2.9)$$

$$\nabla_W I_{i,j} \equiv I_{i,j-1} - I_{i,j} \quad (2.10)$$

and the conduction coefficients are

$$c_{N,i,j}^t = g\left(\left|\nabla_N I_{i,j}^t\right|\right) \quad (2.11)$$

$$c_{S,i,j}^t = g\left(\left|\nabla_S I_{i,j}^t\right|\right) \quad (2.12)$$

$$c_{E,i,j}^t = g\left(\left|\nabla_E I_{i,j}^t\right|\right) \quad (2.13)$$

$$c_{W,i,j}^t = g\left(\left|\nabla_W I_{i,j}^t\right|\right) \quad (2.14)$$

Two functions used by Perona and Malik [67] for $g(\cdot)$ with some constant K are

$$g(\nabla I) = e^{-(\|\nabla I / K\|^2)} \quad (2.15)$$

and

$$g(\nabla I) = \frac{1}{1 + \left(\frac{\|\nabla I\|}{K}\right)^2} \quad (2.16)$$

High contrast edges are favored over low contrast edges in (2.15); while, (2.16) favors wide regions over smaller ones. Using either (2.15) or (2.16), the edges tend to remain sharp in the absence of noise. The presence of noise influences Perona and Malik technique since noise-related gradients are not distinguished from edge-related gradients. In Chapter 3, we expand the basic diffusion equation to contain a scalar function based on the edges of the original image, alleviating this problem.

2.5 Summary of Literary Survey

Barnsley's iterated function system, IFS, theory provides the mathematical background for fractal image compression. Fractal compression algorithms that are based on partitioned or local IFS theory include fractal block coding (Jacquin's method), orthogonal basis iterated function systems, the Bath fractal transform, and hierarchical iterated function systems.

Most of these algorithms use partitioning to maintain a level of accuracy in the fractal mappings. The accuracy of the mapping dictates the quality of the attractor of the fractal code. Because of the limitation of defining images in terms of a small set of affine transformations, coders partition larger regions into smaller regions with small block sizes in the fractal mapping to minimize error. Due to ease of implementation, many coders still use the quad-tree-partitioning scheme.

The major computational complexity in terms of time is the search for the optimal pairs of range and domain blocks for the fractal transformations. The computational complexity in finding these fractal transformations in many coders restricts them from being used in 'real time' applications. Current research into tree structure searching algorithm shows significant reduction in time. Using r-tree data structures for spatial searches reduces the search time down to seconds.

In the next chapter, we examine the mathematical basis for IFS and apply this theory to create a fractal block coder. We show the results of implementing two essential elements needed for a gray scale fractal coder. The two essential elements are a metric based quad-tree partitioning and r-tree data structure for searching. In addition, we look at the sources of error in the attractor which lead to the incorporation of the diffusion techniques into our fractal system for error reduction. By examining these issues, we establish the foundation for our research.

CHAPTER III

3 FRACTAL BLOCK CODERS

Fractal block coders are based on an iterated function system. Iterated function system is a collection of contractive affine transformations that maps any source image onto a desired target image. By modeling these digital images as mathematical spaces, fractal block coding techniques represent digital images as sets of contractive image transformations. The collage theorem developed by Barnsley [5] states that the more accurate the description of the image in terms of contractive image transformations, the closer the attractor of local iterated function system approximates the original image. Before proceeding with the results for our fractal block coder, the next section reviews the mathematical basis for iterated function systems.

3.1 Mathematical Basis for Iterated Function Systems

In order to present the contraction mapping principle, the basic principle in iterated function systems theory, several important concepts need to be defined. This section presents the mathematical concepts of *metric spaces*, *affine transformations*, and *contraction mappings* as defined in [5]. Barnsley [4] and Peitgen [66] give a more extensive background of fractal geometry and techniques.

In fractal geometry, digital images are objects belonging to a complete metric space. A complete metric space is a set or space in which distance can be measured

between any two members of the set and a limit belonging to the set exists for every Cauchy sequence in the set.

Definition [5]: A metric space (\mathbf{X}, d) is a space, or a set, \mathbf{X} together with a real-valued function $d: \mathbf{X} \times \mathbf{X} \rightarrow \mathfrak{R}$, which measures the distance between a pair of points x and y in \mathbf{X} . Suppose that d has the following properties:

$$(i) \quad d(x,y) = d(y,x), \quad \forall x, y \in \mathbf{X}. \quad (3.1)$$

$$(ii) \quad 0 < d(x,y) < \infty, \quad \forall x, y \in \mathbf{X}, x \neq y. \quad (3.2)$$

$$(iii) \quad d(x,x) = 0, \quad \forall x \in \mathbf{X}. \quad (3.3)$$

$$(iv) \quad d(x,y) \leq d(x,z) + d(z,y), \quad \forall x, y, z \in \mathbf{X}. \quad (3.4)$$

Then d is called a metric on the space \mathbf{X} .

Definition [5]: A sequence $\{x_n\}_{n=1}^{\infty}$ of points in a metric space (\mathbf{X}, d) is called a Cauchy sequence if, for any given number $\varepsilon > 0$, there is an integer $N > 0$ such that

$$d(x_n, x_m) < \varepsilon \quad (3.5)$$

for all $n, m > N$.

Definition [5]: A metric space (\mathbf{X}, d) is complete if every Cauchy sequence $\{x_n\}_{n=1}^{\infty}$ in \mathbf{X} has a limit $x \in \mathbf{X}$.

Transformations are mathematical operations for mapping points in a space to other points either in the same space or in a different space. Affine transformations used in fractal image compression are linear transformations with translation that can scale, rotate, stretch, and skew points in the image space.

Definition [5]: A transformation $w: \mathfrak{R}^2 \rightarrow \mathfrak{R}^2$ of the form

$$w(x, y) = (ax + by + e, cx + dy + f), \quad (3.6)$$

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

The following notations [5] can be used to represent an affine transformation:

$$w(x, y) = w \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = \mathbf{Ax} + \mathbf{T} \quad (3.7)$$

where

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad (3.8)$$

and

$$\mathbf{T} = \begin{pmatrix} e \\ f \end{pmatrix}. \quad (3.9)$$

The Contraction Mapping Theorem is the foundation of iterated function systems. Using this theorem, an iterated function system is a collection of contractive mappings defining a unique attractor. When the contractive mappings are iteratively applied to any initial point in the space, the attractor or fixed point of the iterated function system is the limit point that the mappings will approach.

Definition [5]: Let $f: \mathbf{X} \rightarrow \mathbf{X}$ be a transformation on a space. A point $x_f \in \mathbf{X}$ such that $f(x_f) = x_f$ is called the fixed point of the transformation.

Definition [5]: Let $f: \mathbf{X} \rightarrow \mathbf{X}$ be a transformation on a metric space (\mathbf{X}, d) is called contractive or a contraction mapping if there is a constant, $s, 0 \leq s < 1,$ such that

$$d(f(x), f(y)) \leq s \cdot d(x, y) \forall x, y \in \mathbf{X}. \quad (3.10)$$

Any such number s is called a contractivity factor for f .

Theorem (The Contraction Mapping Theorem) [5]: *Let $f: \mathbf{X} \rightarrow \mathbf{X}$ be a contraction mapping on a metric space (\mathbf{X}, d) . Then f possesses exactly one fixed point $x_f \in \mathbf{X}$, and for any point $x \in \mathbf{X}$, the sequence $\{f^n(x): n = 1, 2, \dots\}$ converges to x_f ; that is,*

$$\lim_{n \rightarrow \infty} f^n(x) = x_f, \quad (3.11)$$

for each $x \in \mathbf{X}$.

Representing a digital image as a collage of smaller images, IFS uses the Contraction Mapping Theorem to define a digital image as an attractor to a set of equations. The local IFS is a collection of local contraction mappings called w_i on a complete metric space denoted by (\mathbf{X}, d) , where \mathbf{X} is the space of discrete domain, finitely bounded, real valued signals (image space) and d is the metric or distance measure.

An image is divided into regions or blocks \mathbf{D}_i . The transformations, $w_i: \mathbf{D}_i \rightarrow \mathbf{X}$, will map the blocks onto the image for $(i = 1, 2, \dots, N)$ where N is a finite positive integer. Let \mathbf{S} denote a subset of \mathbf{X} . Using the IFS theory, a contractive operator W is defined as

$$W(\mathbf{B}) = \bigcup_{i=1}^N w_i(\mathbf{D}_i \cap \mathbf{B}), \quad (3.12)$$

for all $\mathbf{B} \in \mathbf{S}$ such that the attractor \mathbf{A} (decompressed image) of W satisfies the condition

$$\lim_{n \rightarrow \infty} \mathbf{A}_n = \mathbf{A} \quad (3.13)$$

where

$$\mathbf{A}_n = \bigcup_{i=1}^N w_i(\mathbf{D}_i \cap \mathbf{A}_{n-1}) \quad (3.14)$$

for $(N = 1, 2, 3, \dots)$ and

$$\mathbf{A} = \bigcup_{i=1}^N w_i (\mathbf{D}_i \cap \mathbf{A}) = W(\mathbf{A}). \quad (3.15)$$

Comparing Eq. 3.13 to Eq. 3.11, we see that the attractor is the fixed point to a set of contraction mappings defined by blocks in the image. Fractal block coding compresses images by finding a set of local contraction mapping w_i . This set is found by minimizing the error distance of each mapping leading to the target image being approximated by attractor \mathbf{A} of W on the image space.

For the compression of gray scale images [66], the contractive transformation takes the form

$$w_i = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}. \quad (3.16)$$

The coefficients a_i , b_i , c_i , and d_i act on the position of the pixel (x, y) according to one of the symmetries presented in Table 3.1 with the translation (e_i, f_i) . The coefficient s_i scales the intensity of the pixel z controlling the contrast of the block while o_i offset the intensity of the pixel controlling the brightness of the block.

Applying Fisher's optimization [36] with respect to root mean square metric, d_{rms} [61],

$$d_{rms}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}, \quad (3.17)$$

the scale s is

$$s = \frac{n(\sum_{i=1}^n d_i r_i) - (\sum_{i=1}^n d_i)(\sum_{i=1}^n r_i)}{n \sum_{i=1}^n r_i^2 - (\sum_{i=1}^n r_i)^2} \quad (3.18)$$

Table 3.1: Eight symmetries in local contraction transformations.

Symmetry	Matrix	Description
0	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	identity
1	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	reflection in y-axis
2	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	reflection in x-axis
3	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$	180° rotation counter-clockwise
4	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	reflection in line $y = x$
5	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	90° rotation counter-clockwise
6	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$	270° rotation counter-clockwise
7	$\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$	reflection in line $y = -x$

and the offset o is

$$o = \frac{1}{n} \left(\sum_{i=1}^n r_i - s \sum_{i=1}^n d_i \right) \quad (3.19)$$

where r_i and d_i are the intensity values of two blocks containing n pixels. If

$$n \sum_{i=1}^n d_i^2 - \left(\sum_{i=1}^n d_i \right)^2 = 0 \quad (3.20)$$

then s is set to zero and o is

$$o = \frac{\sum_{i=1}^n r_i}{n}. \quad (3.21)$$

These equations provide a mathematical robust representation for fractal image compression. Although the next section presents a simple fixed block fractal coder based on a more simplified representation for the fractal transformation, these mathematical concepts still apply for our fractal coder.

3.2 Fixed Block Fractal Coder

In this section, we implemented a simple fixed block fractal coder. For ease of implementation, we modeled the fractal transformation of each range block R as

$$R = \Gamma_i (s \Delta D_{x,y} + o) \quad (3.22)$$

where $\Delta D_{x,y}$ is a spatially reduced version of the domain block D located at pixel location x and y , s controls the contrast of the block, o controls the brightness of the block, Γ_i represent one of the symmetries in

Table 3.1. For each pixel in $\Delta D_{x,y}$, we averaged four pixels in D creating a half scale version of D . We spatially reduced the domain blocks by half for mapping to a range block.

To encode the image, our coder followed the steps outlined below:

1. Input a digital image with intensities values from 0 to 1.
(Note: The gray levels represented by 0 to 255 have been scaled to values from 0 to 1.)
2. Partition the image into non-overlapping range blocks.
3. Partition the image into overlapping domain blocks.
4. For each range block, search through the transformed domain blocks for the one that minimized error distance of mapping D on to R using Eq. 3.22.
5. Generate the fractal code.

For a 256x256 version of ‘Lena’, the fractal code contained 1026 lines of fractal coefficients (Table 3.2 contains the first five lines.) for range blocks of size 8x8. Each line consisted of x , y , i , s , and o , the coefficients needed for Eq. 3.22.

Table 3.2: Partial listing of fractal code.

x	y	Symmetry	s	o
25	97	6	0.04	0.49
41	77	3	-0.13	0.60
97	117	4	0.02	0.50
5	85	1	-0.58	0.82
73	69	0	-0.32	0.69

Since we processed the range blocks sequentially, we decoded the image by applying Eq. 3.22 in the same order. The steps we took to decode the image were as followed:

1. Initialize an image to a constant gray level of .5.
(Note: The gray levels represented by 0 to 255 have been scaled to values from 0 to 1.)
2. Partition the image into non-overlapping range blocks.
3. Partition the image into overlapping domain blocks.
4. To create each new range block, apply Eq. 3.22 to the domain block located at x and y using the corresponding values for s and o .
5. Bound the intensity values of the new range block to between 0 and 1, inclusively.
6. Repeat steps 2 through 5, until the image converges.

For example, for the first range block in the image of 'Lena', the coder fetched the spatially reduced domain block starting at pixel locations, $x = 25$ and $y = 97$. The coder multiplied the pixel intensity values of the spatially reduced domain block by 0.04 and added 0.49 to the intensity values. The coder rotated the pixels in the block clock-wise by 270 degree. The coder bounded intensity levels to values from 0 to 1 by simply clipping the intensity levels outside these values to 0 or 1. The coder replaced the intensity values of the first range block with these new intensity levels. This sequence is applied for the entire list and repeated until convergence.

Figure 3.1 shows the convergence of iteratively applying the fractal code for three test images: 'Lena', 'Goldhill', and 'camera man'. Results in Figure 3.1 shows that the attractor of the fractal code usually converged in less than five iterations. For comparison, Figure 3.2 shows the attractors of the coder for range block size 8x8 and 16x16 and the quantitative measures are contained in Table 3.3 for these sizes.

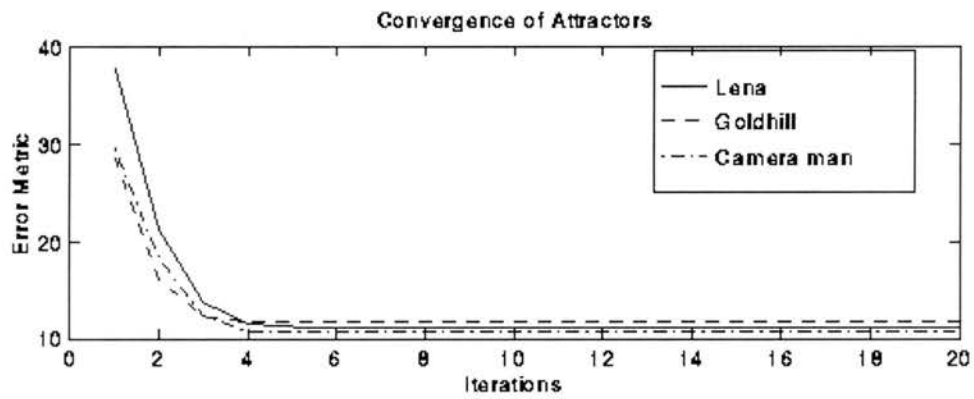


Figure 3.1: Iterations of the fractal code



(a) Attractor of 8x8 range size. (b) Attractor of 16x16 range size.

Figure 3.2: Attractor of fractor code for different range block size.

Table 3.3: Results for using 8x8 and 16x16 range sizes.

Range size	Compression Ratio	PSNR	ξ
8x8	17.65	27.2	0.57
16x16	70.62	23.3	0.39

These results show the relationship range block sizes have on the compression ratio, PSNR, and the edge preservation measure. Compression ratio was four times higher for the 16x16 range block than the 8x8 range block but PSNR and ξ were significantly lower. Smaller block sizes increase image fidelity at the cost of lower compression ratios.

In order to optimize the trade off between compression ratio and image fidelity, we implemented quad-tree partitioning in our fractal coder to determine the maximum size of the range block based on a fidelity criterion in Section 3.4. Before presenting this implementation, we examined the error distance and PSNR in the attractor for a reduced version of 'Lena' in the next section.

3.3 Error Due to Inaccurate Mappings

For examining error distance and PSNR in the attractor, the fixed block fractal coder uses only 8x8 range block size with no partitioning scheme. In addition, the target image is a reduced version (128x128 pixels) of 'Lena' in order to provide a higher concentration of details. The higher concentration of details emphasizes the errors in the attractor of fractal block coding. Figure 3.3 illustrates the attractor of the fractal code or approximation of the target for range block size of 8x8 for the reduced version of 'Lena'. In comparison to the attractors in Figure 3.2, we notice that the errors in the attractor in Figure 3.3 are more pronounced.

In addition, the coder uses a brute force search technique to search the library blocks for a domain block that meets an error tolerance criterion. Instead of searching the entire set of domain blocks, the search will terminate when error distance is less than the error tolerance level. If it does not find a domain block that meets this requirement after searching the entire set of domain blocks, it will choose the domain block that resulted in the least error distance. The error distance is the root mean square metric,

d_{rms} , defined by (3.17). Using an error tolerance of 0.001 allows reduction of search time while still maintaining a high degree of fidelity.



Figure 3.3: Comparison of 'Lena' (128x128) with the attractor.

Since the target image of 'Lena' is 128x128, with 8 bpp (bits per pixel) image, the number of bits in the original image is $128 * 128 * 8$ or 131072 bits. With fixed range blocks of 8x8, a 128x128 target image requires 256 transformations to represent it; therefore, the number of bits in the fractal code is $256 * 29$ or 7424 bits. The compression ratio for the image in Figure 3.3 is 17.65. The PSNR for the attractor in Figure 3.3 is 22.86 dB. Comparing these values to the results for the 8x8 range block size in Table 3.3, PSNR drops from 27.2 to 22.86. This drop in PSNR corresponds to the reduction of the image from 256x256 to 128x128. Higher concentration of details in the 128x128 image leads to inaccurate mappings as shown later in this section.

Visually inspecting images in Figure 3.3 shows the loss of details in the decompressed image. Areas containing texture like the feathery region of the hat suffer

the most while smooth areas suffer the least. In terms of the loss of edges, areas of strong straight edges have less distortion than areas of weak edges.

The error metric measures the error distance between the range block and the mapping of the domain block. Figure 3.4 shows the PSNR for each block related to the accuracy of the mapping the domain block onto the range block. In general, the smaller error distance resulted in higher PSNR for the range block in the attractor of the fractal code.

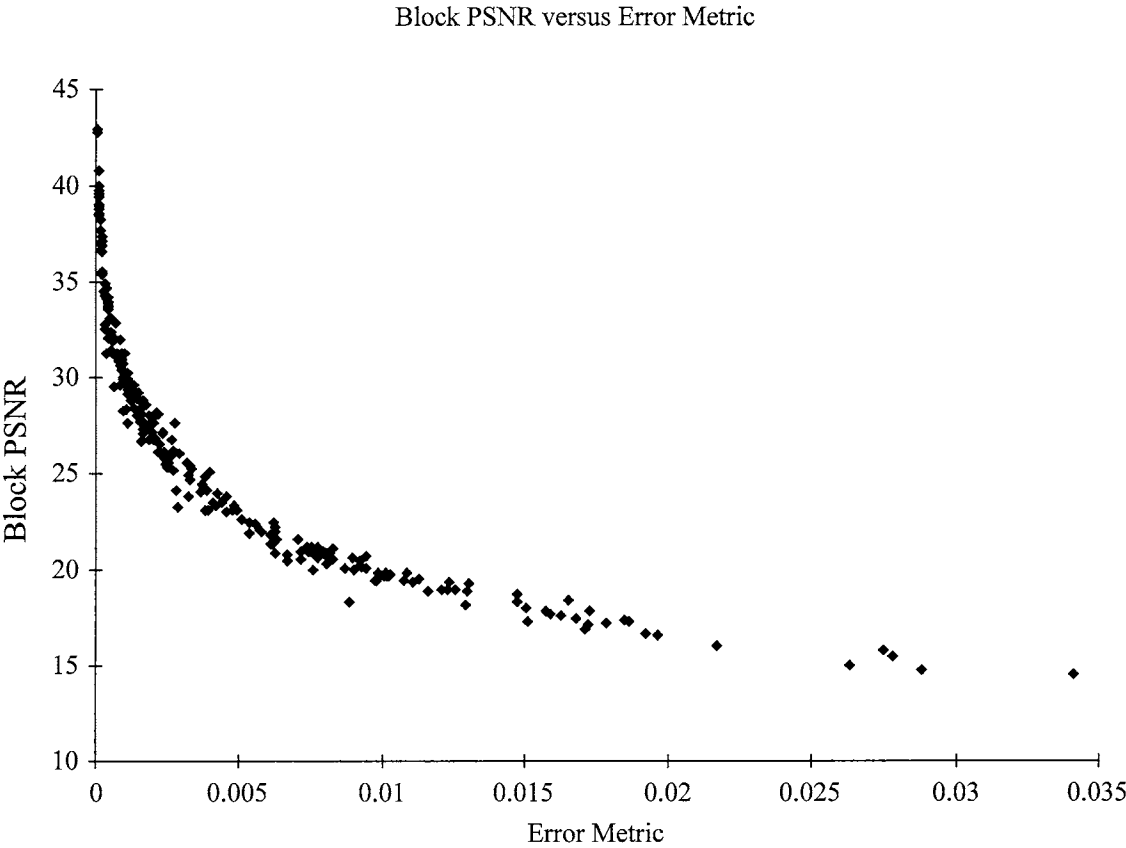


Figure 3.4: Block PSNR versus Error Metric.

Figure 3.5 categorizes the blocks in the original image into three images depending on the PSNR of the related blocks in the attractor of the fractal code. Figure 3.5b contains blocks of the original image that were reconstructed with a PSNR greater than 30. Figure 3.5c contains blocks that were reconstructed with a PSNR between 20 and 30 inclusively. Figure 3.5d contains blocks that were reconstructed with a PSNR below 20. Quantitatively, Figure 3.5 shows that smooth regions are reconstructed with the least amount of distortion while high detailed regions are reconstructed with the most amount of distortion.

In order to see the relationships of the error metric and PSNR in the different regions of the original image, Figure 3.6 contains calculated images of the PSNR and the error metric. In these images, a corresponding gray level between 0 and 255 replaced the numerical values. In the PSNR image, Figure 3.6c, the white regions represent the highest PSNR values while the black regions represent the lowest values. In Figure 3.6b, the error image, the darker regions represent higher error distances. Counting from the lower left-hand corner of Figure 3.6b, the grid box with the highest error is six boxes to the right and six boxes up. In the same location in Figure 3.6a, this area contains three different textural regions, hair, feathers, and part of the brim of the hat. In the same location in Figure 3.6c, the grid box shows a low PSNR value.

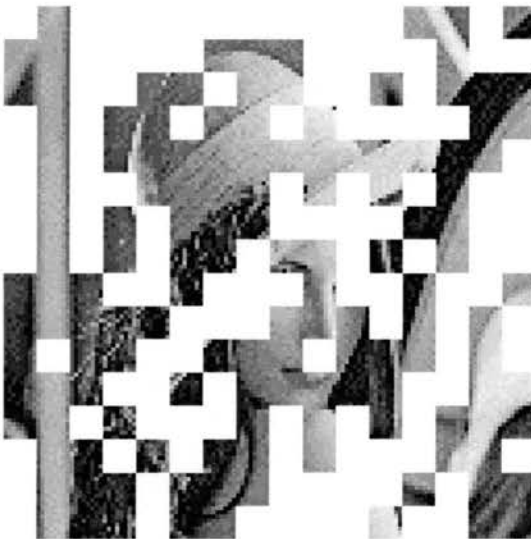
Figure 3.7 shows the residual error image created by subtracting the attractor from the original image and scaling to fit within the image space. The residual error shows that the major portion of errors occurs at edge locations. The accuracy of the attractor of the fractal code depends on the degree of similarity between the larger to smaller blocks mapping. Due to the limitation of the fractal transform mappings, we see loss of edge information, discontinuity at boundaries and blocking artifacts in the attractor at high compression ratio. These degradations and spurious artifacts limit fractal coders.



(a) Original Image



(b) Block PSNR > 30

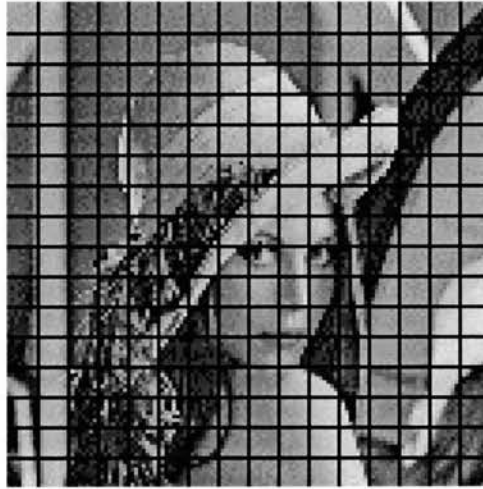


(c) $20 \leq \text{Block PSNR} \leq 30$

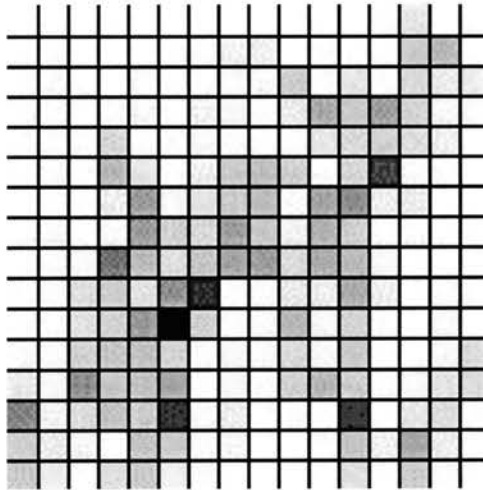


(d) Block PSNR < 20

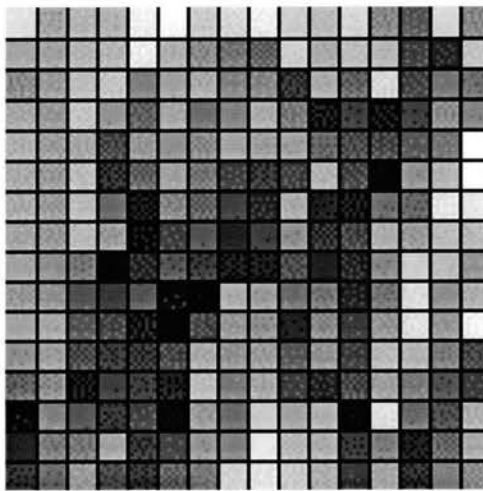
Figure 3.5: Classification of the original image into areas of PSNR values.



(a) Original Image



(b) Error Metric Image



(c) PSNR Image

Figure 3.6: Representations of the error metric and PSNR for image blocks.



Figure 3.7: Residual error image.

In order to increase the degree of similarity, the next section implements a metric based quad-tree partitioning similar to the one proposed by Jacquin [35]. Although, use of smaller blocks sizes reduce the loss of detail, it is important only to use these smaller blocks when needed. Each smaller level of partitioning decreases the compression ratio or increases the bpp (bits per pixel rate) by a factor of four for that region.

Partitioning increases the probability of restoring edges in the attractor; however, it does not insure the retention of edges. Even with smaller block sizes, degradation of edges and loss of edge information occur in the attractor due to the limitation of fractal code to describe the similarities in the image with contractive affine transformations. In addition, the iterative nature of spatially reducing larger regions to smaller regions tends to “flatten” or “blur” the attractor even more [46]. Finally, the independent processing of

each block by the fractal transform causes blocking effects in terms of artificial block boundaries and discontinuities between adjacent blocks.

The next section presents the results of the implementation of a metric quad-tree partitioning similar to the partitioning used by Jacquin to compensate for the error distances. In the next chapter, we address the problem of restoring lost edge information.

3.4 Metric Based Quad-tree Fractal Coder

Based on earlier works of Jacquin and Fisher [36][18], implementation of a quad-tree partitioning improves the basic fractal technique. Quad-tree partitioning provides a simple method to determine the areas that need smaller blocks. The overhead of using quad-tree partitioning is small since it only requires one bit extra per transformation. Based on Table 1.1, the bits required for each transformation increases from 29 for a fixed block coder to 30 for a quad-tree coder. However, the impact on compression is significant. With the quad-tree method, compression ratio in a region decreases by a factor of four when that region needs to be partitioned to maintain a level of accuracy. In order to reconstruct that region, the fractal code requires four additional lines of fractal code instead of one. Assuming each line in the fractal code requires 29 bits [20], covering an 8x8 pixel image block with 8 bpp (bits per pixel) results in a compression ratio of 17.65 or bpp rate of 0.45. Splitting this block into four 4x4 pixel blocks causes the compression ratio to be reduced to 4.4 or a bpp rate of 1.8.

The coder starts with the largest range block size. It searches through the corresponding set of domain blocks or library for the domain block that meets the error tolerance criterion. If a domain block according to that size cannot be found with an error tolerance criterion, the range block is divided into four equal size sub-blocks. This process is repeated until either a domain block is found meeting the criterion or the range blocks have reached the minimum allowed block size. For this analysis, the fractal coder

incorporated only a two level quad-tree partition. The target image is a reduced version (128x128) of 'Lena'. The fractal coder uses two range block sizes of 8x8 and 4x4. The coder uses the error tolerance level to determine if the 8x8 block needs to be spilt into four 4x4 blocks.

In Figure 3.8, a significant improvement in image fidelity is seen in the attractor of the fractal code at an error tolerance level of 0.001. The cost of increasing the image fidelity to this level (PSNR = 25.86 dB) was the reduction of the compression ratio to 5.5:1. From the images in Figure 3.8, rapid degradation of the image occurs as the error tolerance level is increased. The plots in Figure 3.9 and Figure 3.10 give the impact on PSNR and compression for the different error tolerance levels. Lowering the error tolerance level does increase image fidelity measured by PSNR at the cost of lowering compression ratio. Figure 3.11 graphically shows the cost of PSNR versus compression ratio. If a straight line approximates PSNR in Figure 3.11, its slope is approximately -0.63. Using this approximation, the cost of increasing the compression ratio by one would result in a drop of 0.63 dB in PSNR.

Increasing the size of the image to the more standard 256x256 or 512x512 pixels would greatly increase fidelity since the library search space is greater and the details in the image are less concentrated [20]. For the larger images, the quad-tree level can be expanded to include larger block sizes, thus increasing the compression ratio. For example, in a four level quad-tree structure, a 32x32 image block can be divided into a combination of 16x16, 8x8, and 4x4 image blocks. Increasing the size of the image increases the search time due to the increase in the number of range and domain blocks. In the next section, r-tree data structures will be used to reduce the search time.



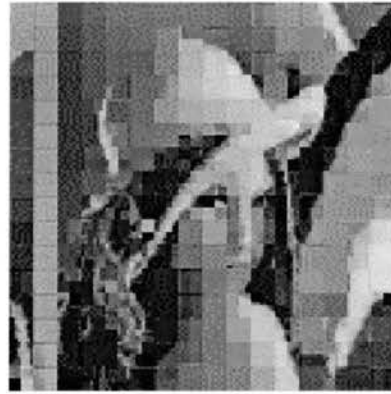
(a) Original Image



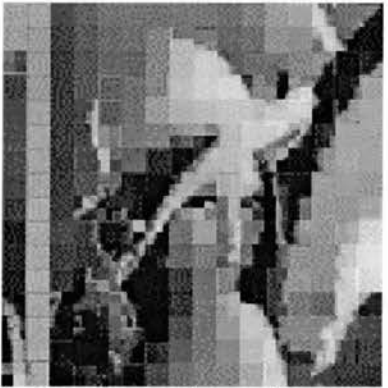
(b) Reconstruction at level = .001
PSNR = 25.86 dB
Compression Ratio = 5.5 : 1



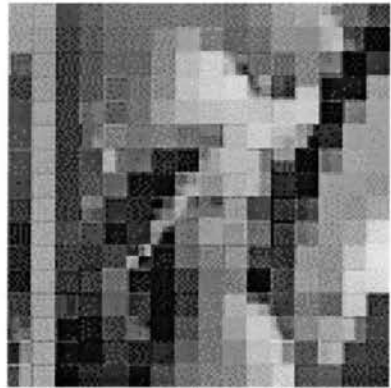
(c) Reconstruction at level = .005
PSNR = 23.56 dB
Compression Ratio = 8.5 : 1



(d) Reconstruction at level = .007
PSNR = 22.48 dB
Compression Ratio = 9.4 : 1



(e) Reconstruction at level = .01
PSNR = 21.47 dB
Compression Ratio = 11.5 : 1



(f) Reconstruction at level = .03
PSNR = 19.15 dB
Compression Ratio = 16.9 : 1

Figure 3.8: Comparison of the attractor at different error tolerance levels.

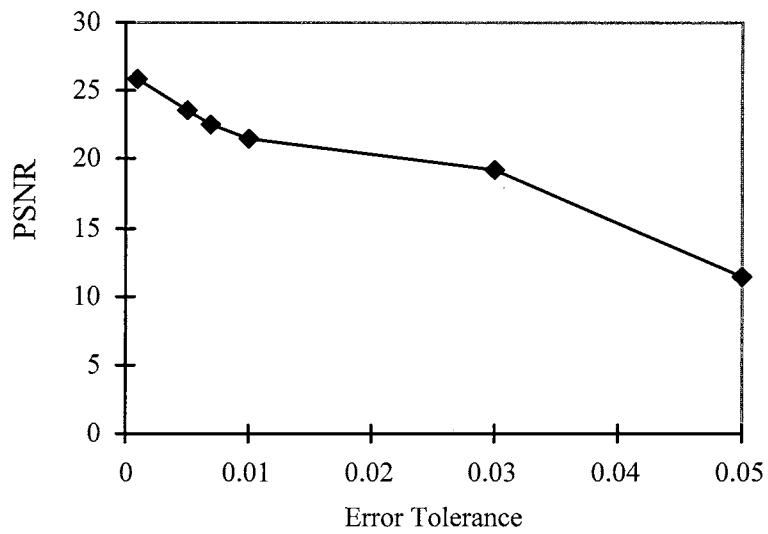


Figure 3.9: PSNR versus error tolerance for metric based quad-tree coder.

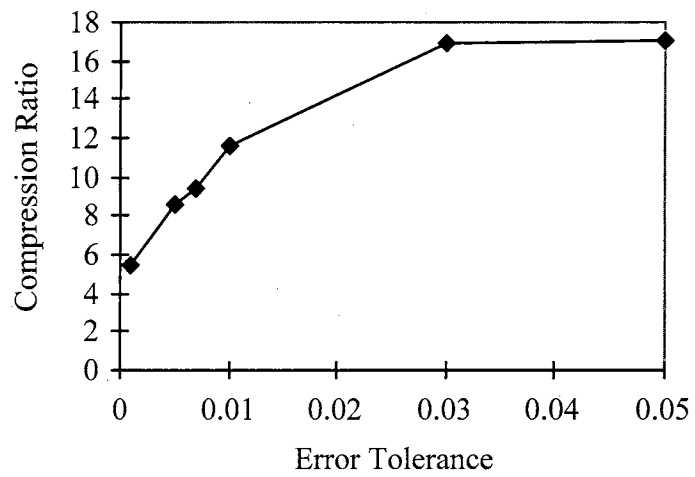


Figure 3.10: Compression versus error tolerance for a metric based quad-tree coder.

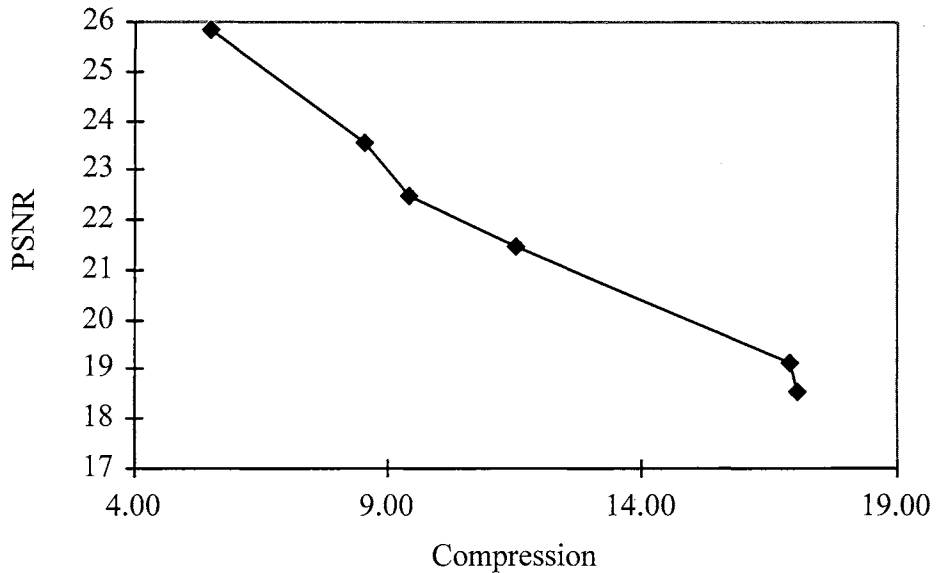


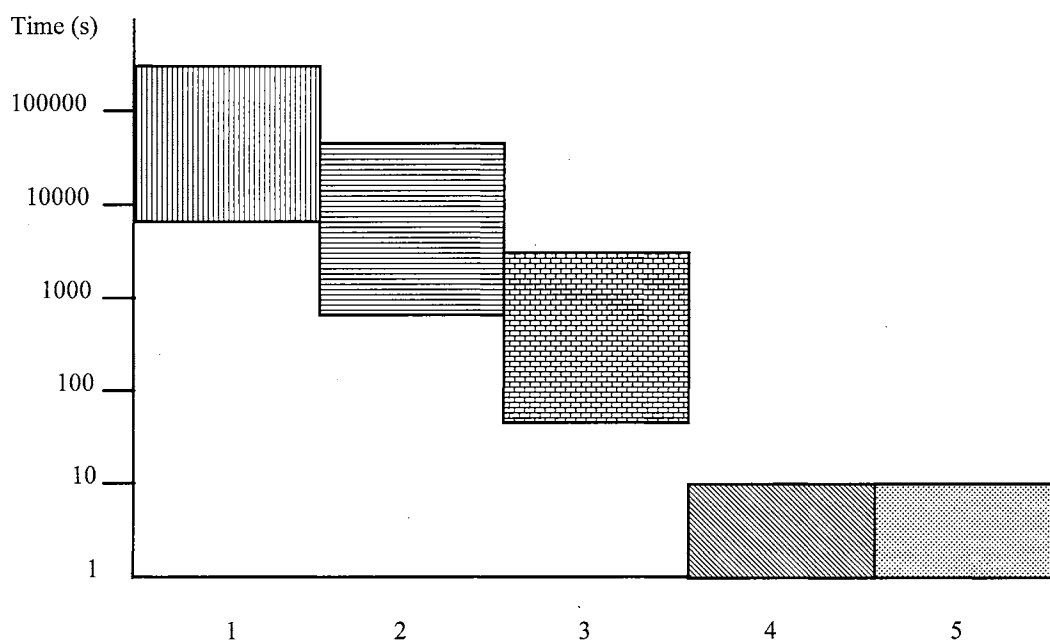
Figure 3.11: PSNR versus compression for a metric based quad-tree coder.

3.5 Fast Fractal Image Compression Using R-trees.

One major obstacle to the widespread use of fractal image compression is the prohibitive CPU time needed in the compression stage. Searching for ‘optimal’ pairs of similar blocks in the image constitutes the major computation load for fractal coders. R-trees overcome this obstacle by reducing search time to seconds. R-tree is a dynamic multi-dimensional index data structure capable of fast retrieval of data objects based on their spatial locations.

The search for this optimal pairing can produce an enormous computational load. For example, an image size of $2^n \times 2^n$ using four rotations and four reflections can create $[(2^n - 2^m + 1)^2 * 8]$ different $2^m \times 2^m$ domain blocks and $(2^n / 2^p)^2$ different $2^p \times 2^p$ range

blocks. Numerous pair comparison of domain and range blocks can produce search time from minutes to days for full search or brute force coders depending on the size of the image and its associated search space. Figure 3.12 illustrates the encoding times compiled by Kominek [40] for different fractal coders. The time it takes to compress an image ranges from seconds to days.



Legend:

1. Even on a fast PC, brute force algorithms take over a day to compress an image.
2. Light brute force algorithms use a simple scheme to reduce the computational load. Compression takes 1-5 hours, times typical of the academic programs.
3. Iterated Systems' commercial software reduces the times to the range of 3-30 minutes.
4. Iterated Systems also sells add-in hardware boards that bring the compression times down to the range of 1-10 seconds.
5. With the recent development of the Fast Fractal Image Compression (FFIC) algorithm, an all-software method now exists capable of compressing images in 1-10 seconds.

Figure 3.12: Encoding time for different fractal coders.[40]

Kominek [41] uses r-trees, dynamic multi-dimensional index data structures, in his Fast Fractal Image Compression (FFIC) method to reduce the time to seconds for 4x4 range block size. This section examines the Kominek's FFIC method. Although we implemented this method for 4x4 blocks, the size of the r-tree structure at large block sizes increases the difficulties of implementation into our fractal system. In the next chapter, we enhance this method to simplify the implementation.

The Fast Fractal Image Compression, FFIC, algorithm uses the pixel values directly to compose the position vector in the indexed multi-dimensional space. Therefore, a 16 dimensional position vector represents a 4x4 image block. In order to use the r-tree structure to search, we need to normalize the pixel values in all the blocks to a fixed mean and variance. Normalization of the vectors establishes a common for valid comparison between range and domain blocks.

Mathematically, we form a vector x ,

$$x = (p_1, p_2, p_3, \dots, p_{16}) \quad (3.23)$$

from the 4x4 image block X ,

$$X = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \\ p_{13} & p_{14} & p_{15} & p_{16} \end{bmatrix}, \quad (3.24)$$

consisting of pixel intensities p_i . The coder creates the position vector y by normalizing the vector x such that the mean of y , μ , is equal to a constant c_1

$$\mu = \frac{1}{n} \sum_{i=1}^n y = c_1 \quad (3.25)$$

and the variance, σ^2 , is equal to a constant, c_2

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (y - \mu)^2 = c_2. \quad (3.26)$$

The r-tree algorithm organizes the domain position so nearby vectors are similar in the position space. The r-tree structure arranges the domain position vectors in a nested manner. Due to the r-tree structure, the coder examines only a few of the domain data objects need to find domain data object that minimizes the error distance for any range data objects. The average search time [41] for his method is approximately proportional to $a \log_b(a)$ where a is the number of domain data objects and b is the maximum data objects bounded by any rectangle.

Using pixel values to define the position vectors, block sizes of the order $2^n \times 2^n$ would require 2^{2n} dimensional position vectors to define its spatial location. Clearly, larger block sizes increase the complexity of the r-trees requiring more time and memory to process. Since fractal compression relies on using large blocks for high compression ratios, we need to develop a more effective method for spatial location representation of the blocks.

3.6 Summary of Current Methods

Fractal coders compress digital images by relating areas of local self-similarity at different scales in the images. In an image, numerous smaller regions look like larger regions in the image; for example, a small cloud looks like a large cloud. Taking advantage of this correlation, the fractal code pairs up similar regions creating a list of fractal mappings. The attractor of this code approximates the original image. The closeness of the attractor to the original image depends on similarity between the larger to smaller mapping.

Searching for 'optimal' pairs of similar blocks in the image constitutes the major computational load for fractal coders. Numerous pair comparison of domain and range blocks can produce search time from seconds to days for full search or brute force coders depending on the size of the image, its associated search space, and search algorithms.

To overcome long encoding time, FFIC uses r-trees, dynamic multi-dimensional index data structures, to reduce the time to seconds for 4x4 range block sizes. Due to the spatial representation of the image block in this method, the efficiency decreases and the complexity of the r-tree search algorithm increases for larger block sizes.

In order to increase the degree of similarity, fractal coders use quad-tree splitting to partition the image into different block sizes. The quad-tree splitting method divides a large square or block into four equal sized smaller squares. In smooth areas of the image, using the larger block size gives higher compression ratios. In areas of edges, the smaller block size increases the accuracy of the fractal mapping. Compression ratio in a region decreases by a factor of four when that region is partitioned down to maintain accuracy. In order to maintain high compression ratio, the coder needs to minimize partitioning down to smaller blocks.

Partitioning increases the probability of restoring edges in the attractor; however, it does not insure the retention of edges. Degradation of edges and loss of edge information occur in the attractor due to the limitation of fractal code to describe the similarities in the image with contractive affine transformations. In addition, the iterative nature of spatially reducing larger regions to smaller regions in the attractor for image reconstruction tends to “flatten” or “blur” the image even more.

In the next chapter, our research focuses on overcoming some of the limitation of fractal block coding. This is necessary in order to achieve a fast edge preserving fractal system for ‘real-time’ applications like surveillance.

CHAPTER IV

4 A FAST EDGE PRESERVING FRACTAL SYSTEM

This chapter focuses on developing a fast edge preserving fractal system for use in ‘real-time’ identification applications like surveillance. Fractal coders perform well at high compression ratio and thus are suitable for these applications; however, long encoding time and loss of edge information are major drawbacks. Tradeoffs between compression ratio, edge preservation, and encoding time are the major concerns in the development of a fast edge preserving fractal system (FEPFS).

Overcoming these limitations to produce FEPFS is the objective of this research. To meet the objectives, the design criteria for FEPFS for compressing 256x256 images are as follow:

1. Compression ratios are above 40:1.
2. Encoding times are less than one minute.
3. The attractor contains significant edge information.

Before proceeding the achievements of FEPFS in respect to the design criteria, the next section presents the iterative error compensation technique. Presentation of the iterative error compensation technique developed during the initial stages of our research for Sandia National Laboratories gives the foundation for the design criteria.

4.1 Iterative Error Compensation Technique

This simple algorithm shows the possibility of using fractal image compression in surveillance applications. We develop the iterative error compensation technique to improve the attractor of the fractal code of a fixed block fractal coder using large sizes and just the identity symmetry. This method is an initial attempt to compensate for the loss of edges at large block sizes for high compression ratio and to decrease encoding time for satellite field surveillance.

Figure 4.1a is an original 256x256 image of a military vehicle provided by Sandia National Laboratories for testing. The calculation for the compression ratio is different in this section due to the simplification of the fractal transform and the addition of an error term. We calculate the compression ratio using the actual file size of the fractal code; therefore, compression ratio is equal to the size of the original file divided by size of the fractal code.

Figure 4.1b is the attractor of the fractal code using an 8x8 domain block size resulting in a compression ratio of 5.49:1 (12.47 for entropy coded file using PKZIP). Figure 4.1c is the attractor of the fractal code using a 16x16 domain block size resulting in a compression ratio of 21.72:1 (40.74:1 for entropy coded file using PKZIP).

For most applications, the image or attractor produced in Figure 4.1c is not acceptable. For the purpose of surveillance, Figure 4.1b is acceptable since trained personnel from Sandia National Laboratories could identify the military truck.

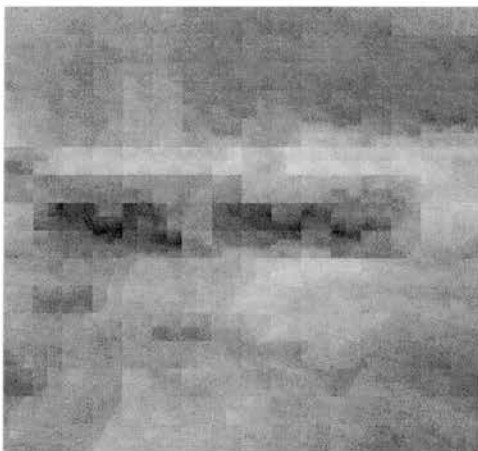
We simplify the fractal transform (3.22) by removing the scale term and allowing only the identity symmetry. Simplifying the algorithm, limiting the search space, and using large block sizes reduce the compression time from days to minutes for 256x256 images. Using the large blocks increases the compression ratio and shortens the compression time at the cost of attractor quality as shown in Figure 4.1.



(a) Original



(b) At 8x8, compression ratio is 12.4.



(c) At 16x16, compression ratio is 40.7.

Figure 4.1: Results for different block sizes in fractal block coding.

In Figure 4.1c, the details such as the windows of the military vehicle were lost in the attractor of the fractal code using the 16x16 block size. Adding error compensation to the fractal transform compensates for the reconstruction errors and restores the details of the original image to the attractor of the fractal code. The iterative error compensation technique is a unique method of restoring high frequency components in the attractor of the fractal code. Figure 4.2 is a flowchart of the technique, as it fits into the fractal scheme.

First, the coder partitions the image into non-overlapping range blocks. From the set of domain blocks created from the image, the coder determines the best mapping for the first range block. The coder forms an error block of 0's and 1's according to a threshold criterion. The coder finds an error gain term that minimizes the L_2 error measure between the range block and its associated domain block. The coder outputs the parameters of mapping transformation and the error block. The coder repeats the process for every range block. The coder iteratively applies the mapping and the error block in the fractal code until the attractor converges.

From a theoretical viewpoint, the technique uses the error blocks E to minimize the L_2 , the least squares, distance measure between the fractal transformation R

$$R = \Gamma(\Delta D_{x,y} + o) + gE \quad (4.1)$$

and the corresponding range block in the original image. The terms in (4.1) are defined as

$\Delta D_{x,y}$ is a spatially reduced version of the domain block,

$D_{x,y}$ is located at pixel location (x,y) ,

o is the difference in the intensity means of original range and $\Delta D_{x,y}$,

Γ represents the identity symmetry in

Table 3.1, and

g is a gain term for optimization of the error blocks.

For each pixel in $\Delta D_{x,y}$, we averaged four pixels in D creating a half scale version of D .

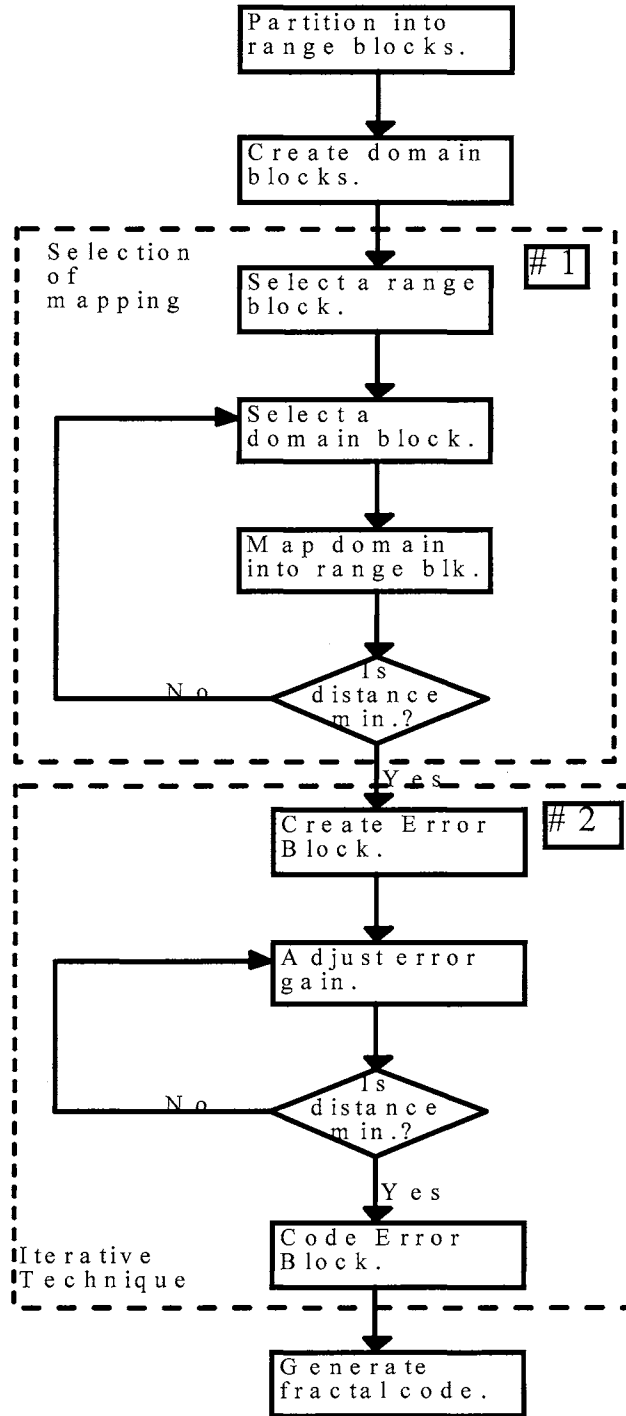


Figure 4.2: Flowchart of the iterative error compensation technique.

Initially, for each range block, the coder searches for the fractal transformed “best fit” domain block as shown in section #1 of Figure 4.2. After choosing the domain block, the coder determines E and g to minimize the error distance of the fractal transform (4.1) as shown in section #2 of Figure 4.2. The closeness of R to the original range block dictates the convergence of the attractor to the original image. The error term Eg in (4.1) gives us an addition parameter to optimize the accuracy of the contractive fractal mapping.

The error blocks, E , consist of entries, E_{ij} ,

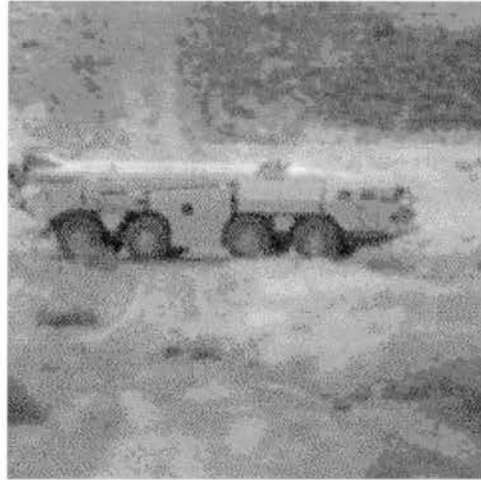
$$E_{ij} = \begin{cases} 0; R_{ij}^{orig} > R_{ij} \\ 1; R_{ij}^{orig} < R_{ij} \end{cases} \quad (4.2)$$

where the subscript ij represents the pixel intensity values in the original range blocks and the transformed domain blocks. Representing the error information with two symbols, (0,1), allows the compensation method to be simple and efficient.

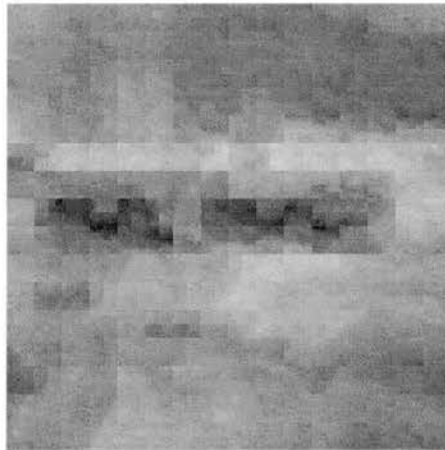
The coder represents the error blocks by index of the error library generated statistically for maximum error compensation. The coder outputs only two addition bytes for representing the error information for each block. Integration of the error term into the fractal mapping creates a more accurate description of the original image.

To decode the attractor of fractal code, the coder iteratively applies the error term as part of the fractal transformation to reconstruct a more detailed quality image. As can be seen by Figure 4.3c, applying (4.1) on a target image causes the target image to converge to an approximation of the image. In this manner, the error compensation technique takes advantage of the iterative nature of fractal image compression.

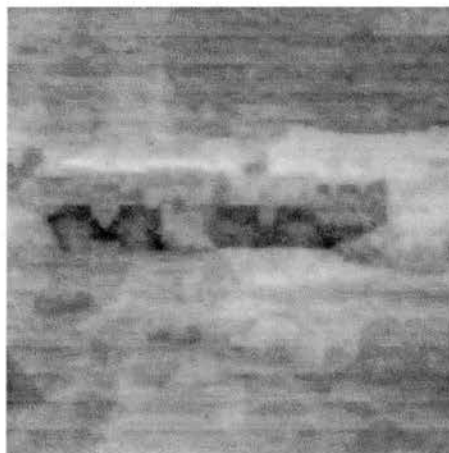
Results for 16x16 block size show considerable improvement in the details of the image in the attractor of the fractal code in Figure 4.3c than in Figure 4.3b. Error compensation restores the windows of the vehicle in Figure 4.3c. Figure 4.3 shows that the primary object of interest, the military truck, is more detailed in the attractor with error compensation.



(a) Original Image



(b) Attractor of the fractal code with No Error Compensation



(c) Attractor of the fractal code with Iterative Error Compensation

Figure 4.3: Results of the iterative error compensation technique.

Use of the error compensation technique improves the image of the military truck significantly. These improvements allow Figure 4.3c to be acceptable for target recognition in our surveillance application. The results show that the iterative error compensation technique improves the image quality of the attractor of the fractal code at large range block sizes. It restores the details of the original image in the attractor of the fractal code and, at the same time, maintaining a compression ratio of approximately 40:1. This technique allows fractal block coding to be used with large block sizes in order to gain the necessary compression ratio.

Although error compensation method restores the details of ‘interest’ for this application, it requires additional processing and storage in terms of the error blocks. In addition, it increases the complexity of the coder.

Based on the research done during this initial phase, we set our design criteria for a fast edge preserving fractal system, FEPFS. The fractal system with iterative error compensation proves that compression ratios above 40:1 are possible for level of fidelity set by Sandia National Laboratories. The next goal for FEPFS is to reduce the search time to seconds.

4.2 Wavelet R-tree Search for Fractal Coders

The wavelet r-tree search for fractal coders uses a new method to represent spatial locations of image blocks. This method generates the spatial locations for indexing the image blocks from the normalized lower order coefficients of the wavelet transform. Based on the Daubechies wavelets, the wavelet transform we used represents the intensities in the image block more compactly.

In comparison to current methods, implementation of the wavelet r-tree method reduces the dimensions of the search space. Lower dimensions lead to reductions in computational complexity, search time, and memory requirements. The wavelet r-tree

search engine proves to be an efficient versatile search engine for fractal coders at minimal cost to the quality of the image.

Instead of using the pixel values, the proposed wavelet r-tree search uses the lower order coefficients of the wavelet transform of the image blocks to generate the position vectors, creating a more efficient r-tree structure. Since wavelet transforms concentrate the energy in the lower order coefficients [72], the lower order coefficients reflect the majority of the blocks' feature information. The variance of the lower order coefficients is much higher than the variance of the higher order coefficients. For example, for the 8x8 block wavelet coefficients of a 256x256 version of 'Lena', the variance of the lower order coefficients is 0.0588 compared to 0.0027 for higher order coefficients. Therefore, the low frequency coefficients have the greatest impact on the spatial location comparison between the domain and range blocks.

The wavelet r-tree search averages neighboring pixel values in the domain blocks to reduce the larger domain blocks to the size of the range blocks. This is necessary to produce the same dimensional search space. The algorithm calculates the wavelet coefficients for each domain block. Because the fractal transform affects the scale and offset of the image blocks, the normalization of the wavelet coefficients between 0 and 1 allows valid comparison of the position vectors for the fractal transform.

The algorithm eliminates the higher order wavelet coefficients, thus reducing the dimensions of the position vector. It uses the normalized lower order coefficients of the wavelet transform to create the position vector for indexing the fractal parameters of each block. We modified the r-tree to insure that a domain data object is found for every range position vector.

Figure 4.4 and Figure 4.5 show the results of testing the wavelet r-tree search in a fractal coder on a Power Macintosh 6100/66. The fractal coder uses 16x16 domain and 8x8 range blocks to compress a 256x256 version of 'Lena'. Figure 4.4 shows the effect on the peak-signal-to-noise (PSNR) of eliminating the wavelet coefficients in generating

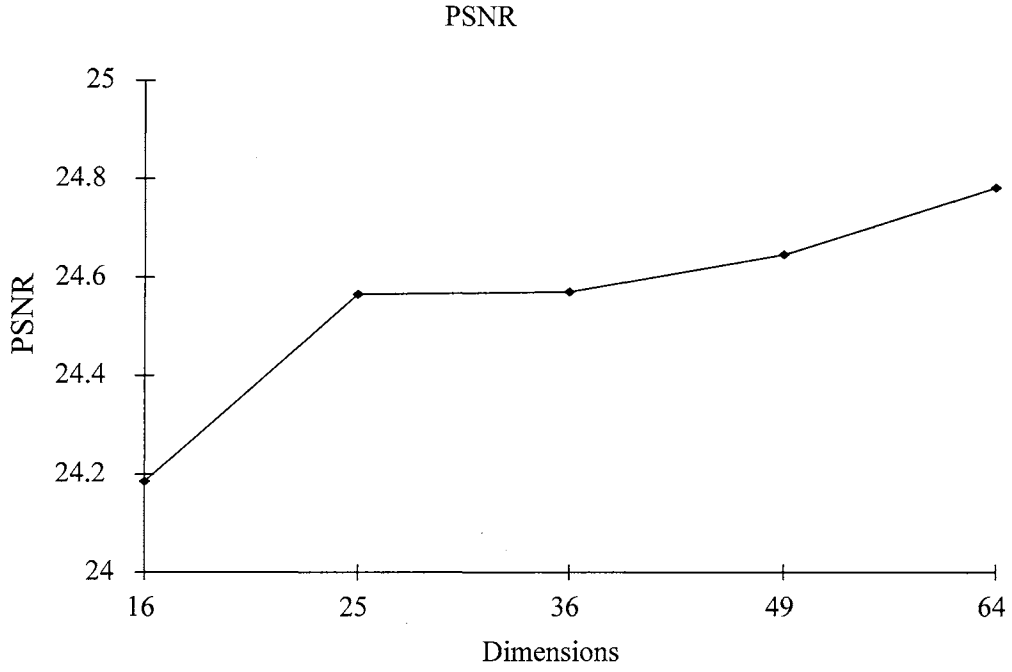


Figure 4.4: PSNR versus dimensions.

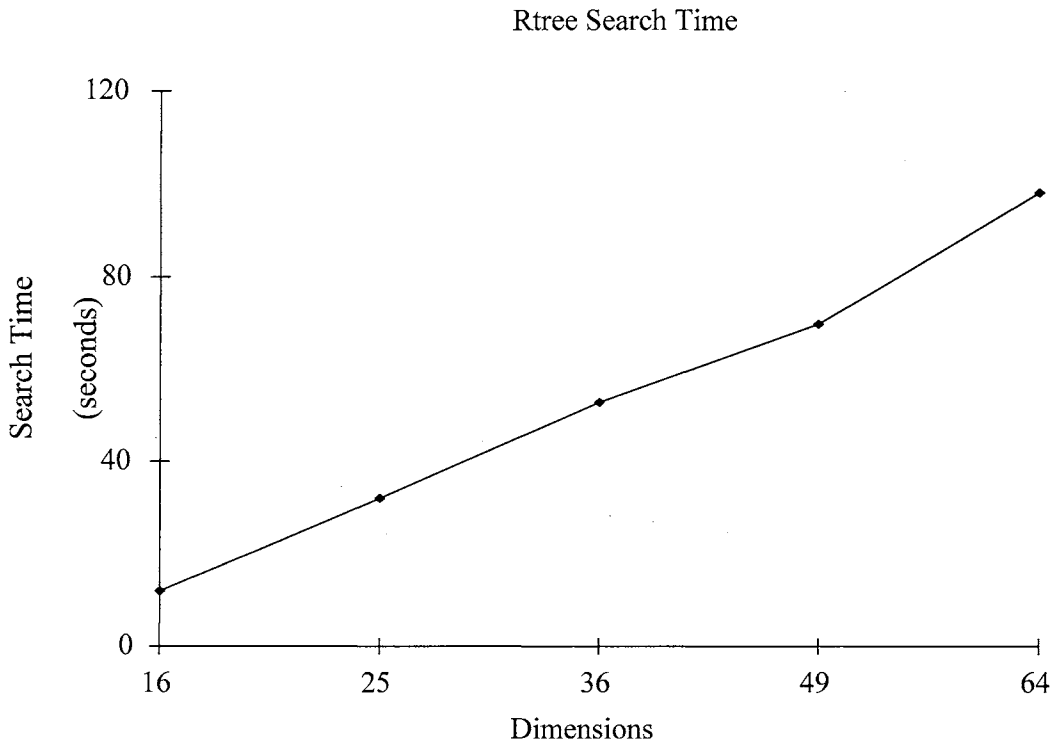


Figure 4.5: Search time versus dimensions.

the position vector. Using all 64 wavelet coefficients to create the position vectors gives a PSNR of 24.8 dB comparable to using all 64 pixel values in the blocks to create the position vectors. Reduction of the dimensions of the position vector from 64 to 16 by eliminating the higher order coefficient results in only a 0.6 dB drop in the PSNR.

This approach increases the speed of the r-tree search time by a factor of 8.1 reducing the search time from 98 to 12 seconds, which is shown in Figure 4.5. It also decreases the memory storage requirement of the r-tree significantly. For example, the reduction from 64 to 16 position vector of floats (4 bytes typical for floats) resulted in a decrease from 256 to 64 bytes for each domain block. Due the large number of domain blocks used in the r-trees, the memory saving can be significant in the implementation of the r-trees. For this test, the memory saving is roughly 28Mbytes.

Using only the low frequency wavelet coefficients to characterize the spatial location of image blocks creates an efficient search engine for fractal coders. Unlike using the pixel values, it increases the versatility of the usage of r-trees for larger block sizes without significantly increasing the dimensions of the search space. It speeds up the search for ‘optimal’ pairing of similar blocks in the image by reducing the complexity of large dimensional search space. It reduces the memory storage requirement for implementing the r-trees.

Using the wavelet r-tree search, FEPFS reduces its search time for compressing 256x256 images to seconds. By meeting its second design goal, FEPFS has the compression speed necessary for near ‘real time’ applications.

4.3 Edge Based Quad-tree Partitioning Method

To lessen the impact of the effects on compression, increase the preservation of edges, and provide a front-end loader for our search engine, we developed an edge based quad-tree fractal coder. In Figure 3.7, the error image shows that the loss of edges is a

significant part of the distortion in the attractor of the fractal codes. Considering this information, we change the splitting criterion in the quad-tree algorithm from the conventional error metric to an edge-based criterion.

For this analysis in this section, the fractal coder incorporated only a two level quad-tree partition. The target image is a reduced version (128x128) of 'Lena'. The edge-based two level quad-tree fractal coder uses the edge content of the block to determine if the 8x8 block needs to be split into four 4x4 blocks.

By using the edge information the partitioning of the image is independent of the fractal mapping unlike the metric based quad-tree method. For an 8x8 block in 128x128 image, before a full search coder partitions the 8x8 block into four 4x4 blocks, it has calculated the error metric for all 102,152 domain blocks. The error metric calculation requires 64 subtractions, 64 multiplications, and 63 additions. In this worst case scenario, the full search coder performs 19,511,032 math operations before splitting the 8x8 block into 4x4 blocks. Since the edge-based quad-tree method is independent of the fractal mapping, the image blocks are split based on the edge content of the image blocks. The edge content of the image is calculated initially before the fractal encoding process begins. If the edge content of the range block exceeds the edge threshold, the image block will be split. For an 8x8 block, this requires 63 additions before the block can be split. It performs this operation only once.

The coder uses the Sobel operators [28] to detect the vertical and horizontal edges in the image. It detects edges from the magnitude of the gradient vector. The gradient vector points in the direction of maximum rate of change. Using a threshold value, the coder calculated the edge images for the reduced version of 'Lena' in Figure 4.6. For each range block, adding the pixel values of the combined edge image gives the edge content. If the edge content exceeds a predetermined edge splitting level, the coder divides larger block into smaller blocks.

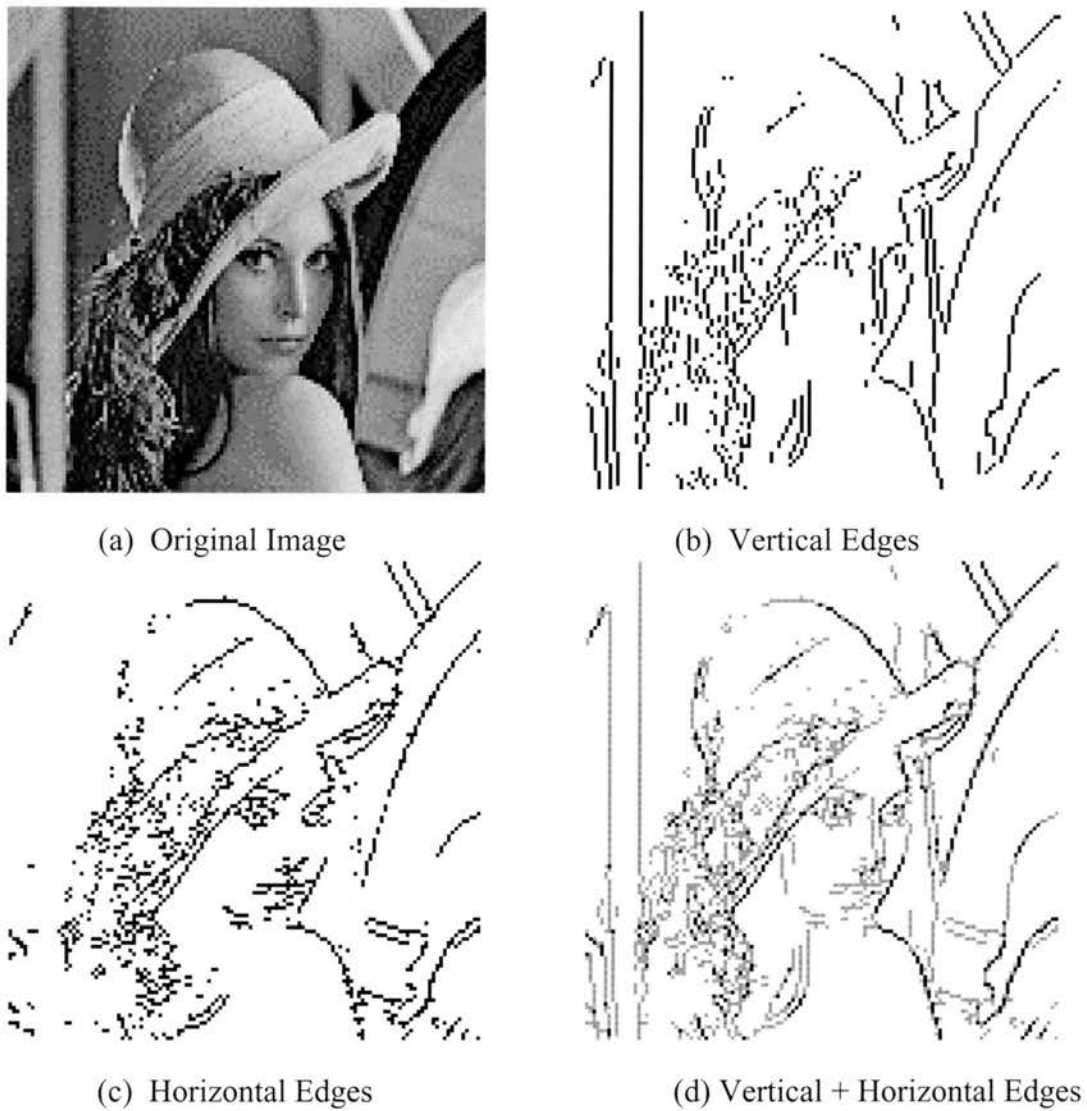


Figure 4.6: Edge images for 'Lena'.

From the images in Figure 4.7, the degradation of the image occurs as the edge splitting level increases. Visually, this degradation occurs at a lesser rate than the metric based quad-tree partition. Figure 4.8 gives the impact on PSNR for the different edge splitting level at two given error tolerance levels, 0.001 and 0.005. Figure 4.9 shows how the edge splitting level affected the compression ratio.



(a) Original Image



(b) Splitting level = 5
PSNR = 26.32 dB
Compression Ratio = 5.5.



(c) Splitting Level = 15
PSNR = 25.31 dB
Compression Ratio = 8.0.



(d) Splitting level = 25
PSNR = 22.47 dB
Compression Ratio = 12.7



(e) Splitting level = 30
PSNR = 23.20 dB
Compression Ratio = 13.8 : 1



(f) Splitting level = 35
PSNR = 23.05 dB
Compression Ratio = 15.1 : 1

Figure 4.7: Comparison attractor at different edge splitting level.

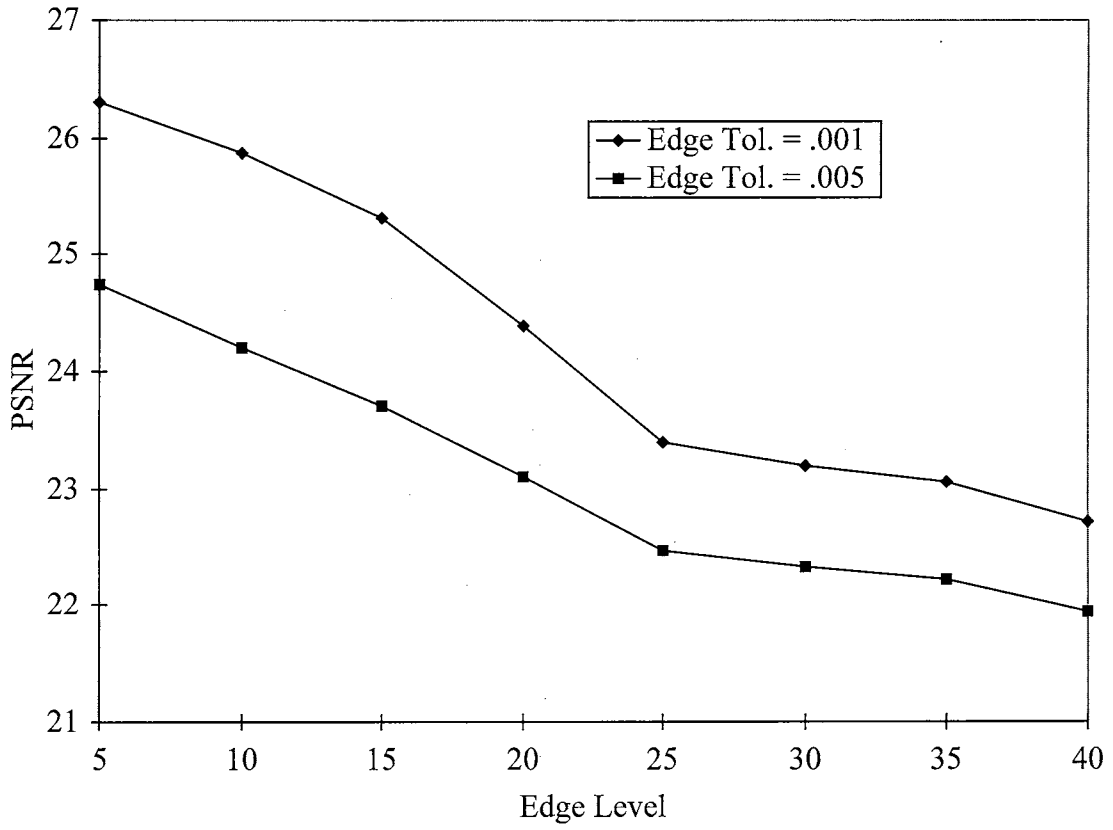


Figure 4.8: PSNR versus edge level in edge based partitioning.

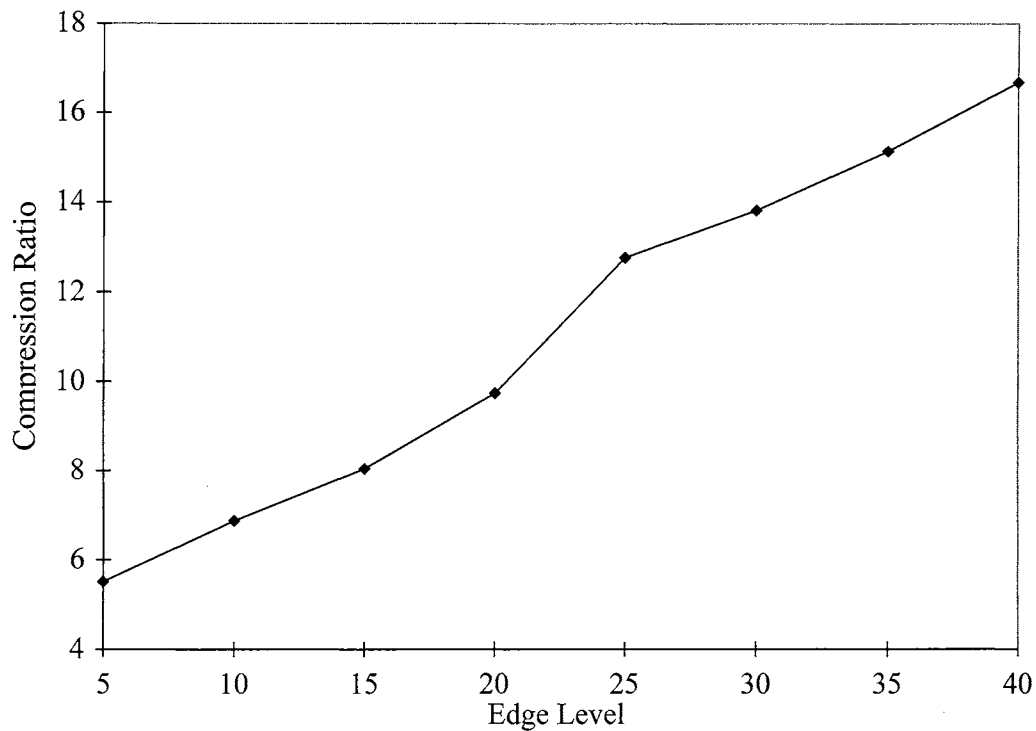


Figure 4.9: Compression versus edge level in edge based partitioning.

With this partitioning scheme, compression does not depend on error tolerance but PSNR does. Figure 4.8 shows using lower error tolerance improves PSNR values. Figure 4.10 graphically shows the cost of PSNR versus compression ratio for two different error tolerance levels. Approximating the PSNR in Figure 4.10 by a straight line gives a slope of -0.32. Using this approximation, the cost of increasing the compression ratio by 1 decreases the PSNR by 0.32 dB for the edge based quad-tree fractal coder. This cost has been reduced by a factor of 2 in comparison with the cost of 0.63 dB for the metric based quad-tree fractal coder.

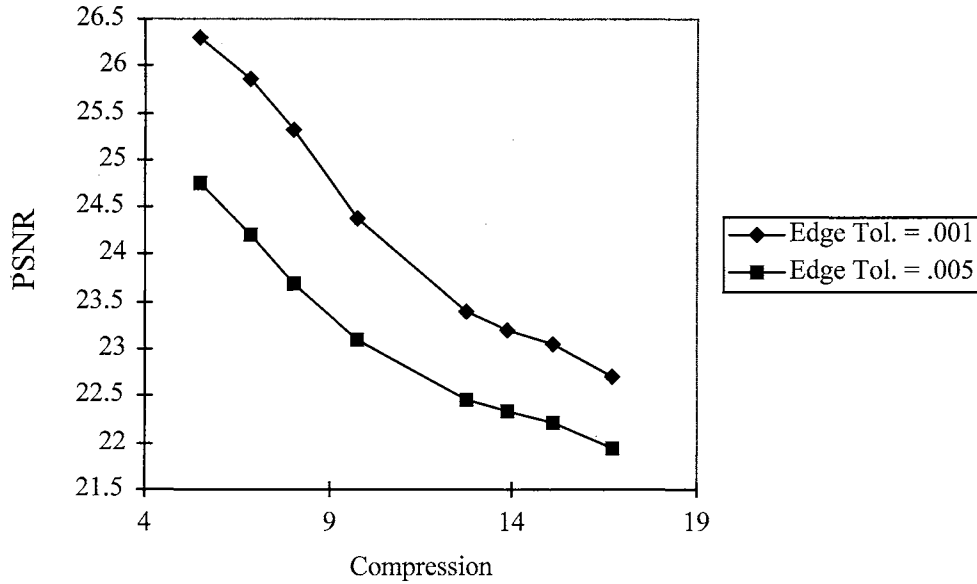


Figure 4.10: PSNR versus compression for edge based partitioning.

For comparison, Figure 4.11 shows the attractor for of the fractal codes for a compression ratio of approximately 9.5 for fixed, metric-based, and edge-based coders. The edge-based fractal coder outperforms the metric based quad-tree fractal coder in terms of image fidelity to compression ratio tradeoff. The edge based coder preserves the edges better at higher compression ratios.

Edge based quad-tree partitioning determines the edge regions needing smaller blocks totally independent of the fractal equations. Determining the regions needing smaller block size according to the edges in original image eliminates fruitless time-consuming searches in larger block sizes library.

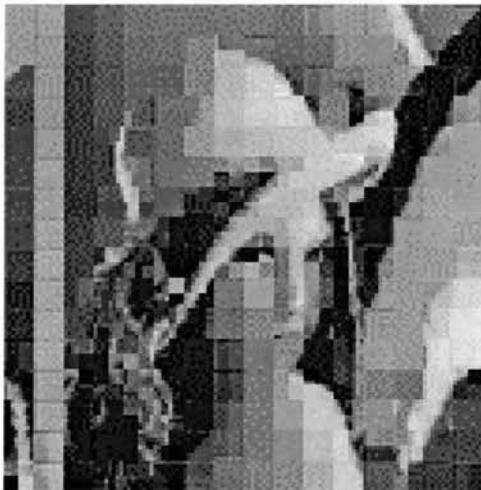
In addition, the edge-based coder provides a front-end loader for the wavelet r-tree search engine. Knowing optimal block size for each image region, encoding the image simply requires the coder to retrieve the optimal domain block for range from the corresponding r-tree data structure.



(a) Original Image



(b) Fixed Blocks of 8x8



(d) Metric Based Quad-tree



(c) Edge Based Quad-tree

Figure 4.11: Attractors of different partitioning methods.

Inaccurate fractal mappings with large image block sizes increase losses of edge information, discontinuities at boundaries and blocking effects. Partitioning the large block into smaller blocks reduces the losses of edge information in the attractor. Fractal block coding process each range block independently of the neighboring range blocks and block coding does not correlate across the boundaries. Therefore, partitioning does contribute to discontinuities at the block boundaries and blocking effects.

Although there are many techniques used to overcome the blocking effects and the discontinuities [47],[68], they do not in general restore lost edge information. Diffusion techniques provide a method for restoring edges in addition to smoothing noisy edges, discontinuities, and blocking artifacts. In the next section, we show how diffusion techniques overcome these problems and preserve significant edge information at a lower bit rate cost than partitioning down to smaller block sizes.

4.4 Edge Preserving Diffusion Techniques

Our diffusion technique uses backward diffusion to restore edges and forward diffusion to reduce the artifacts and smooth noisy edges. Expanding the basic diffusion equation to contain a scalar function, we direct the diffusion process to smooth along the direction of edges and sharpen in the direction of edges. In this manner, we can restore edge information and smooth discontinuities and blocking effects.

4.4.1 Mathematical Development

Diffusion techniques model the intensity of the image f as an anisotropic diffusion [76] in the form of

$$\frac{df}{dt} = \nabla \bullet (k \nabla f(x, y, t)) \quad (4.3)$$

where $\nabla \bullet$ and ∇ define the divergence and gradient operators. Letting the diffusion constant k vary, we control the direction and rate of the diffusion spatially. Substituting k in (4.3) by $g(x, y, t)$ gives

$$\frac{df}{dt} = \nabla \bullet (g(x, y, t) \nabla f(x, y, t)). \quad (4.4)$$

Since f and g are scalar functions, using Green's theorem [39], (4.4) reduces to

$$\frac{df}{dt} = g(x, y, t) \nabla^2 f(x, y, t) + \nabla g(x, y, t) \nabla f(x, y, t) \quad (4.5)$$

where ∇^2 defines the Laplacian operator. Allowing $f(x, y, t)$ to be a twice differentiable scalar function, (4.5) can be rewritten as

$$\frac{df}{dt} = g(x, y, t) \nabla \bullet \nabla f(x, y, t) + \nabla g(x, y, t) \nabla f(x, y, t) \quad (4.6)$$

since

$$\nabla^2 f(x, y, t) = \nabla \bullet \nabla f(x, y, t). \quad (4.7)$$

The diffusion process can be modeled as an evolution of curves embedded in higher dimensional surfaces [42]. Representing the gray intensity level $z = I(x, y)$ as a surface S given by

$$S : f(x, y, z) = z - I(x, y) = 0, \quad (4.8)$$

implicitly defines the surface S . The implicit function theorem [18] implies that the surface defined by $f(x, y, z)$ always exists and the differential of $f(x, y, z)$ does not vanish.

By definition [39], the normal vector of S at point P is given by the gradient of f at point P and the divergence of this gradient gives a vector that is orthogonal to the normal vector. In order to retain maximum edge information, we need to smooth along the direction of the edge and not across the edge [42]. Based on this information, (4.6) can be used to control the diffusion process. We want to forward diffuse in the direction given by the divergence of the gradient of f , i.e. the direction given by the first term of (4.6). In order to backward diffuse or sharpen, we want to diffuse parallel to the normal vector represented by the second term in (4.6) with a negative coefficient, i.e. in the opposite direction. To implement (4.6) for edge restoration, we want a scalar function that backward diffuses only at the edge locations.

4.4.2 Using the Original Image as a Scalar Function in Diffusion Equation

In our first attempt to define a scalar function, we chose the scalar function in (4.5) to be the original image. Choosing this scalar function causes our diffusion coefficient to be a function of the gradient of the original image and the gradient attractor. We also have the relationship that the attractor approximates the original. Ideally, as we approach convergence, the attractor should approach the original image.

Applying scalar function in (4.6) allows us to forward diffuse in the direction of Laplacian of the original image (orthogonal to the edge). Also, it allows us to backward diffuse in the direction of the gradient of the attractor modified by the gradient of the image (normal around edge location and zero otherwise).

In order to minimize the cost in terms of compression ratio and simplify the implementation we assumed the following:

- Fractal coding degrades the edges in a manner similar to diffusion; therefore, we can restore the attractor by iteratively applying the changes dictated by (4.6).
- For images, the gradient image is similar to the Laplacian image; therefore, $\nabla f \approx \nabla^2 f$. Visually inspecting these images in Figure 4.12 shows this to be a reasonable assumption for our application since the edge content of both images are correlated. Note: For the Laplacian image, zero crossings represent edge locations; however, in both images, large swings in value represent edge information.



Laplacian Image



Gradient Image

Figure 4.12: Laplacian and gradient images of Lena.

Applying (4.6), we implemented the discrete diffusion in the following manner:

$$g(x, y, t + 1) = g(x, y, t) + \alpha[g(x, y, t)\nabla^2 f(x, y, t) + \nabla g(x, y, t)\nabla^2 f(x, y, t)] \quad (4.9)$$

where α is the percent change in the cross correlation between the Laplacian of the attractor and the image. In (4.8), we reduce the additional information that is needed to be sent with the fractal code to just the Laplacian of the original image.

Since the Laplacian of the attractor g also approximates the Laplacian of the image f , only the residual code r given by

$$r(x, y) = \begin{cases} 1; & \text{if } \nabla^2 f - \nabla^2 g > T \\ -1; & \text{if } \nabla^2 f - \nabla^2 g < T \\ 0; & \text{otherwise,} \end{cases} \quad (4.10)$$

where T is an error threshold set by the user. For example, the residual information for ‘Lena’ shown in Figure 4.13 is calculated using $T = 0.18$. This nomenclature allows us to reduce additional information and to code the information with a smaller number of bits. Using structure of the fractal code to handle 4x4 image blocks, we simply embed the non-zero residual information in 4x4 blocks into fractal code.

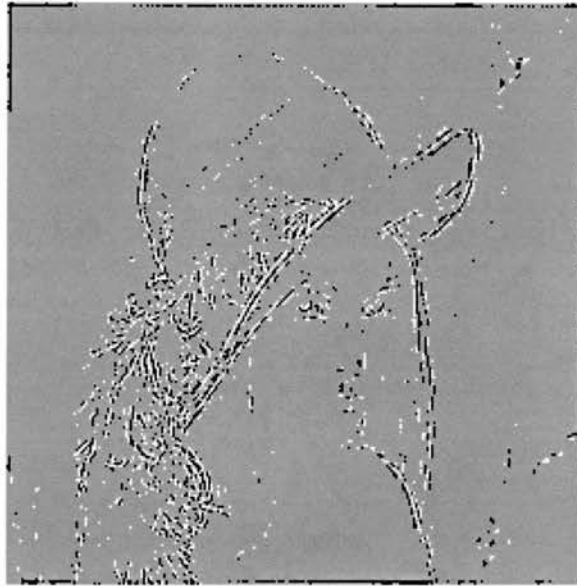


Figure 4.13: Residual image for diffusion.

Figure 4.14 shows the results of the diffusion process for a 256x256 gray level version of 'Lena', validating the usage of (4.8) for image restoration. An approximation of 'Lena' from a simple fixed 8x8 block fractal coder, Figure 4.14a, shows image degradation in terms of blurring, some loss of edge information, and reconstruction artifacts. For clarification, we refer to attractor as the decoded image before applying any post processing or diffusion. We restore the attractor at different threshold values based on the standard deviation of intensity values in the Laplacian image. PSNR is 26.3 dB and ξ is 0.81 for the restored image using the full Laplacian image (Figure 4.14b) in comparison to 26.4 dB and 0.55, respectively. Because PSNR value only gives general indication of image quality and not edge preservation, the changes in PSNR values are not significant. There is a significant increase in ξ from the restoration of the edges.



(a) Attactor for fixed 8x8 range block.



(b) Diffused using the full Laplacian image.



(c) Diffused at $T = \sigma^2$.



(d) Diffused at $T = 2\sigma^2$

Figure 4.14: Comparison of attractor diffused at different levels.

The other two images, Figure 4.14c and Figure 4.14d, have approximately the same ξ at around 0.72 with PSNR values around 26.2. However, we transfer less Laplacian information for Figure 4.14d. From our experiments, we found that ninety percent of Laplacian information was zero or near zero for ‘Lena’. Using a 4x4 block for comparison, the average number of bits required would be $0.9*4*4 + 2*0.1*4*4$ or 17.6

bits for sending this information. The bits per pixel (bpp) rate would be 1.1 (17.6/4/4) which is less than the bpp rate of 1.8 for the quad-tree split. In addition, the number of 4x4 residual blocks is roughly from 30 to 90 percent less than the number of 4x4 partitioned image block needed to achieve a certain value of ξ and error threshold set by the user. For Figure 4.14d, we use 316 residual blocks in comparison to 776 image blocks needed when partitioning down to 4x4 block size, roughly a 40 percent reduction.

To reduce the additional information needed for the diffusion process and to simplify the algorithm, we examine a scalar function based on edge information in the next section.

4.4.3 Edge Based Scalar Function

To simplify and automate the diffusion process, we base our scalar function in (4.4) on the edge map of the original image. In this manner, we can limit the backward diffusion process to the significant edges, thus controlling the diffusion process without a user define threshold. Since we wanted the diffusion to occur at only edges, we chose the diffusion coefficient scalar function $G_{x,y}$,

$$G_{x,y} = \alpha(1_{x,y} - E_{x,y}) \quad (4.11)$$

where $E_{x,y}$ is a map of the edges defined by the zero-crossings of the Laplacian operator [3], $1_{x,y}$ is a matrix of ones of dimensions x by y , and α is a scaling constant.

In order to smooth while preserving edges, we implement a surface curvature diffusion local filter kernel similar to El-Fallah and Ford in [1] but we expand it to accommodate the scalar function $G_{x,y}$

$$F_{x,y}^{n+1} = F_{x,y}^n + \sum_{i=-1}^1 \sum_{j=-1}^1 w_{i,j} F_{x+i,y+j}^n - \beta(\nabla G_{x,y} \nabla F_{x,y}^n) \quad (4.12)$$

where

$$w_{i,j} = \begin{cases} w_{0,0} & ; i = 0 \text{ \& } j = 0 \\ \frac{1}{8\|\nabla G_{x+i,y+j,1}\|} & ; \textit{otherwise} \end{cases} \quad (4.13)$$

and β is a scaling constant. Note that $w_{0,0}$ is one minus the sum of all the other weights w_{ij} . We use Sobel operators to create gradients of the attractor, $F_{x,y}$, and surface representation of the edge map, $G_{x+i,y+j,1}$.

Modeling the degradation of the edges in the attractor of the fractal code as a diffusion process, we restore some of the edges by iteratively applying the changes dictated by (4.12). In each iteration, we restore and sharpen some of the edges; as well as smooth reconstruction artifacts, blocking effects and discontinuities.

We minimize the cost of the $E_{x,y}$ by just sending lost edge information $E_{l(x,y,z)}$ given by

$$E_{l(x,y,z)} = E_{x,y} - E_{f(x,y,z)}. \quad (4.10)^S$$

ince the attractor contains a significant amount of the edge information, $E_{f(x,y,z)}$. To further minimize the impact on the compression ratio, we embed the edge information in the fractal code because the position of the pairing within the fractal code gives the pixel reference location for the edges. We send only the 4x4 pixel edge blocks that contain significant information unlike the quad-tree partitioning which requires all four sub-blocks of a partitioned block to be sent.

Using a 256x256 test image of ‘Lena’, the results in Figure 4.15 validate the usage of (4.12) for image restoration. Figure 4.15a is the attractor, approximation of ‘Lena’, of the fractal code and Figure 4.15b is the restored image from applying (4.12). The attractor shows image degradation in terms of blurring, some loss of edge information, discontinuities, and reconstruction artifacts. The restored attractor using the

diffusion technique shows some degree of enhancement and restoration. From Figure 4.15b, we see the following results:

- (1) Smoothing by forward diffusion eliminates some of blocking effect in flat intensity regions like the shoulder and cheek areas.
- (2) Sharpening by backward diffusion restores some of the nostril edge information in the nose region.
- (3) A combination of smoothing and sharpening lessens the visual appearance of the discontinuities in the feather region.

Table 4.1 provides a quantitative evaluation of our experiments; the results of applying (4.12) on the attractor of ‘Lena’ compressed at three different splitting levels given in the first column. We calculate the splitting from the amount of edge information contained in each block and normalize it by the size of the block.



(a) Attractor

(b) Diffused Image

Figure 4.15: Comparison of the attractor (a) and the diffused image (b).

Table 4.1: Comparison of quantitative measures for diffusion.

Splitting Level	With no post processing.			Restored with diffusion.		
	Ratio	PSNR	ξ	Ratio	PSNR	ξ
.3	101.6	19.2	0.29	45.7	19.4	0.34
.2	61.9	20.2	0.31	37.4	20.5	0.37
.1	25.28	24.75	0.51	22.2	24.9	0.55

For evaluation purposes, we maintain the same splitting threshold for the three level of partitioning allowed: 32x32, 16x16, and 8x8 block sizes. Columns 2, 3, and 4 in Table 4.1 give the compression ratio, PSNR, and ξ of the attractor of the fractal code. Columns 5, 6, and 7 in Table 4.1 give the compression ratio, PSNR, and ξ of restoring the attractor with the diffusion technique.

Because of the smoothing involved in the diffusion equation, we see little change in the PSNR values. From our experiences, although smoothing increases the visual appeal of the image to the viewer, it generally decreases the PSNR. Using this method, we see approximately an increase of 10 percent in edge preservation.

The impact of sending the additional edge information is considerably less than using the next lower partitioning level of 4x4 block size as shown in Table 4.2. In Table 4.2, we allow the 8x8 blocks to be split into 4x4 blocks to show the impact on compression ratio. In order to achieve an edge measure of 0.55, the splitting threshold level needs to be between 0.15 and 0.16. The corresponding compression ratios are 12.09 and 13.97.

compression ratio. In order to achieve an edge measure of 0.55, the splitting threshold level needs to be between 0.15 and 0.16. The corresponding compression ratios are 12.09 and 13.97.

In comparison, using the diffusion method to achieve an edge measure of 0.55 reduces the compression ratio to 22.2. The impact of using this method on the compression ratio directly related to the splitting level. Lowering the splitting level increase the amount of edge information contained in the reconstruction image; therefore, less edge information needs to be sent.

Diffusion techniques enhance the attractor of fractal coders. These techniques smooth the artificial block boundaries imposed by the fractal transform. The gradient information of the image allows the image to be smooth along the edges and, thus reducing some of the discontinuities at block boundaries. Using the backward diffusion at edge location restores some of the significant edge information.

Table 4.2: Quantitative measures for splitting level without diffusion.

Level	Ratio	PSNR	ξ
.30	37.8	19.2	0.33
.20	20.0	20.5	0.43
.16	14.0	21.5	0.52
.15	12.1	22.9	0.59
.10	7.7	27.2	0.76

Results in this section show diffusion techniques are viable image enhancement and restoration tools for fractal coders. Our work shows that the diffusion techniques enhance the quality of the attractor by restoring lost edge information and reducing reconstruction artifacts.

FEPFS leverage diffusion techniques to create a more robust image compression system. Using the diffusion techniques, FEPFS increases the edge correlation between the original image and the attractor significantly; therefore, achieving the criterion of edge preservation. FEPFS achieves preservation at a lower bpp rate cost than using the current quad-tree partition method. In the next chapter, we conclude with a summary of the contributions of FEPFS and future areas of research.

CHAPTER V

5 CONCLUSIONS

The focus of this research is the development of a Fast Edge Preserving Fractal System (FEPFS) for 'real time' applications like satellite surveillance. Current fractal image compression schemes suffer from losses of edge information at high compression ratio and long search time due to computational complexity of finding optimal pairings in the fractal mappings. Fractal coders compress digital images by relating areas of local self-similarity at different scales in the image. Taking advantage of this correlation, the fractal code pairs up similar regions creating a list of fractal mappings. The attractor of the fractal code represents an approximation of the original image. Table 5.1 compares a quadtree fractal coder to a JPEG coder and a wavelet coder at compression ratios above 40:1.

In the attractor, there are three major sources of errors: loss of edge information, discontinuity at boundaries and blocking artifacts. These errors are more significant in larger block sizes used to obtain higher compression ratios. In addition, there is no methodology in current fractal coders to insure the retention of significant edges.

Many academic fractal coders take minutes to hours to compress a 256x256 image. Although there is a commercial fractal coder that only takes seconds, it requires specialized hardware to achieve its fast encoding times. These limitations need to be overcome in order to create a viable compression system for 'real time' applications.

In the next section, our contributions on overcoming the limitations of current fractal coders are summarized. Our compression system, FEPFS, compresses 256x256

digital images at compression ratios greater than 40 in less than one minute while minimizing edge discontinuities and blocking effects and restoring some of the significant edge information.

Compression Ratio	PSNR		
	Fractal	JPEG	Wavelet
40	29.3	28.9	30.7
50	28.4	26.5	29.2
60	27.8	<25	28.6
70	26.8	<25	27.7
80	26.4	<25	26.9
90	26.1	<25	26.3

Table 5.1: Comparison of results for 512 x 512 Lena [21].

5.1 Contributions of FEPFS

The fast edge preserving fractal system, FEPFS, provides a practical, viable, robust compression methodology for ‘real time’ application like surveillance. These applications dictate the need for high compression ratio, fast encoding times, and edge preservation. FEPFS obtains compression ratios above 40:1 by using large block sizes and entropy coding the fractal code.

To minimize the distortion in high detailed areas, FEPFS uses a four level edge quad-tree partitioning scheme to determine the optimal size of the blocks. The block sizes will range from 32x32 to 8x8. This allows us to use the smaller blocks sizes along the edges to maximize retention of edge information and minimize the cost of our

diffusion technique. FEPFS uses a diffusion technique to overcome edge degradation and spurious artifacts at large block sizes. It also uses an edge map to control the diffusion technique for restoring some of the lost edge information and insuring the retention of some of the significant edges. Using this technique, FEPFS is able to improve the attractor image quality at lower bit rate cost than partitioning down to 4x4 block sizes. Partitioning down to 4x4 and 2x2 block sizes is the current way that fractal coders try to overcome errors in the attractor.

By expanding the basic diffusion equation to contain a scalar function based on the edges of the original image, FEPFS directs the diffusion process to smooth along the direction of significant edges and sharpen in the direction of the edges. Choosing a scalar function based on the edge map of the original image controls the backward diffusion to the edge location overcoming the problems with backward diffusion in current diffusion techniques. Using this method of diffusion, FEPFS restores edge information and smooth discontinuities and blocking effects.

Searching for 'optimal' pairs constitutes the major computation load for fractal coders. Implementation of the wavelet r-tree search engine in FEPFS reduces the computational complexity, search time, and memory requirements. R-tree data structures provide an efficient heuristic searching mechanism for spatial data. Modifying the basic algorithm of r-tree to search for 'best fit' data objects allows us to use it to search for our fractal pairings. Current fractal coders use the pixel intensity values of the image block to create the tree structure. This method is inefficient and complex for large block sizes. It also does not lend itself to quad-tree partitioning. Unlike these current tree methods in fractal coding, we use the lower order coefficients of the wavelet transform to create our spatial index for our tree. This allows our tree structure to be more compact and efficient resulting in faster search time. In addition, we can create similar tree structures for the different block sizes predetermined by our edge based partitioning method.

In summary, the contributions of the research include:

1. In comparison to full search quadtree fractal coder, FEPFS reduces the encoding times from hours to seconds.
2. With an edge based quadtree partitioning scheme, FEPFS reduces the cost in term of PSNR for increasing the compression ratio by a factor of 2 over standard metric based quadtree partitioning.
3. Using diffusion techniques, FEPFS reduces the bbp rate to 1.8 (using 4x4 blocks) to 1.1. For 256x256 Lena, the compression ratio was increased by a factor of 1.6 for the same edge measure value.
4. Using diffusion techniques, FEPFS restored the edges in 256x256 Lena attractor to an edge measure of 0.81 from an edge measure of 0.55.

With edge based partitioning, diffusion techniques, and wavelet r-tree search engine, we have overcome some of the limitations of current fractal block coders. We have created a viable system for compressing digital images. In the next section, we look at future areas of research for further improvements in fractal coders.

5.2 Future Areas of Research

This research addresses many of the tradeoffs related in creating a working compression system. Our diffusion technique provides a method to preserve edges in large block size mappings in fractal block coding. However, the technique adds additional information to the fractal code and increases the complexity of the fractal system.

In order to minimize the additional edge information, identifying the edge regions that benefit most from this technique would allow us to reduce the edge information. Identification of significant edges is a separate but relevant issue for reduction of edge

information. If we can identify the edges of the target we want to examine, we could send only those edges rather than sending all the edges. This would greatly reduce the amount of additional information and increase compression ratio.

To simplify complexity of implementing the diffusion process, we are striving to create an adaptive diffusion filter based on the local gradient components and statistical properties of the image. We may be able to integrate this filter into the fractal transforms to describe the similarities in the image more accurately.

Currently, we embed the edge information in 4x4 blocks in the fractal code. Since the edge information is binary (consists of only 0 and 1), we need to investigate other compression methods used to compress binary images. Overall, we may find a more efficient method to code the edge information.

REFERENCES

- [1] Baharav Z., Malah D., and Karnin E. "Hierarchical Interpretation of Fractal Image Coding and Its Applications," *Fractal Image Compression: Theory and Application*, Yuval Fisher (editor), pp. 91-117, 1995.
- [2] Bani-Eqbal B. "Speeding up Fractal Image Compression," Internal Report, University of Manchester, 1994.
- [3] Barnsley M.F. "Lecture notes on Iterated Function Systems", *Proceedings of Symposia in Applied Mathematics Vol. 39: Chaos and Fractals The Mathematics Behind the Computer Graphics*, R. L. Devaney and L. Keen (eds.), American Mathematical Society, Providence, RI, pp. 127-144, 1989.
- [4] Barnsley M.F. *Fractal Everywhere*, Academic Press Professional: Boston 1993.
- [5] Barnsley M.F. and Hurd L.P. *Fractal Image Compression*, AK Peters: Wellesley, Massachusetts, 1993.
- [6] Barnsley M.F. and Sloan A. "A Better Way to Compress Images," *BYTE Magazine*, Jan. 1988.
- [7] Barthel K.U., Schuttemeyer J., Voye T., and Noll P. "A New Image Coding Technique Unifying Fractal and Transform Coding," *Proc. ICIP-94*, vol. III, pp. 112-116, 1994.
- [8] Bogdan A. "Multiscale Fractal Image Coding and the Two-Scale Difference Equation," Columbia University Technical Report, TR 358-94-05, pp. 1-14, 1994.
- [9] Bogdan A. "Multiscale (Inter/Intra-Frame) Fractal Video Coding," *IEEE International Conference on Image Processing ICIP '94*, 1994.
- [10] Bogdan A. "The Fractal Pyramid with Applications to Image Coding," *IEEE Proceedings of ICASSP*, 1995.
- [11] Bogdan A. and Meadows H. "Kohonen Neural Network for Image Coding Based on Iteration Transformation Theory," *Proceedings of the SPIE: Neural and Stochastic Methods in Image and Signal Processing*, vol. 1766, pp. 425-436, 1992.
- [12] Bondarendo V.A. and Dolnikov V.L. "Fractal Image Compression by the Barnsley-Sloan Method," *Automation and Remote Control*, vol. 55, no. 5, pp. 623-629, 1994.

- [13] Cheng B. and Zhu X. "Multiresolution Approximation of Fractal Transform," Internal Report, University of Kent, 1995.
- [14] Davis G. "Adaptive Self-Quantized Wavelet Subtrees: A Wavelet-based Theory for Fractal Image Compression," *IEEE Data Compression Conference Proceedings '95*, James Storer (editor), pp. 232-241, 1995.
- [15] Davis G. "A Wavelet-Based analysis of Fractal Image Compression," *IEEE Trans. Image Processing*, vol.7, no. 2, Feb. 1998.
- [16] Davoine F., Bertin E., and Chassery J. "From Rigidity to Adaptive Tessellations for Fractal Image Compression: Comparative Studies," *IEEE 8th Workshop on Image and Multidimensional Signal Processing*, Sept. 1993.
- [17] Dudbridge F. "Fast Image Coding by a Hierarchical Fractal Construction," Internal Report, University of California, 1994.
- [18] El-Fallah A.I. and Ford G.E., "Mean Curvature Evolution and Surface Area Scaling in Image Filtering," *IEEE Trans. Image Processing*, vol. 6, no. 5, pp. 750-753, May 1997.
- [19] Fisher Y. "Fractal Image Compression," *SIGGRAPH '92 Course Notes*, vol. 12, pp. 7.1-7.19, 1992.
- [20] Fisher Y. "Fractal Image Compression with Quad-trees," *Fractal Image Compression: Theory and Application*, Y. Fisher (ed.), Springer-Verlag, New York, 1995.
- [21] Fisher Y. "Comparison of Results," *Fractal Image Compression: Theory and Application*, Y. Fisher (ed.), Springer-Verlag, New York, 1995.
- [22] Fisher Y., Shen T., and Rogovin D. "Fractal (Self-VQ) Video Encoding," *Proceedings of the SPIE: Visual Communications and Image Processing '94*, vol. 2304, 1994.
- [23] Fisher Y., Shen T., and Rogovin D. "A Comparison of Fractal Methods with DCT and Wavelets," *Proceedings of the SPIE: Neural and Stochastic Methods in Image and Signal Processing III*, vol. 2304-16, 1994.
- [24] Forte B. and Vrscay E.R. "Inverse Problem Methods for Generalized Fractal Transforms," Internal Report, University of Waterloo, 1994.
- [25] Forte and Vrscay E.R. "Solving the Inverse Problem for Measures Using Iterated Function Systems: A New Approach," *Advances in Applied Probability*, 1995.
- [26] Gharavi-Alkhansari M. and Huang T.S. "A Fractal-based Image Block Coding Algorithm," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, vol. 5, pp. 345-348, Apr. 1993.

- [27] Gharavi-Alkhansari M. and Huang T.S. "Fractal-based Techniques for a Generalized Image Coding Method," *Proc. ICIP-94*, vol. III, pp. 122-126, 1994.
- [28] Gonzalez R.C. and Woods R.E. *Digital Image Processing*, Addison-Wesley: Reading, MA, 1992.
- [29] Guttman A. "A Dynamic Index Structure for Spatial Searching," *Proceedings of ACM Sigmoid Conference on Management of Data*, pp. 47-57, 1984.
- [30] Hoel E.G. and Samet H. "A Qualitative Comparison Study of Data Structures for Large Line Segment Databases," *Proceedings of ACM Sigmoid Conference*, pp. 205-214, 1992.
- [31] Hurtgen B. "Contractivity of Fractal Transforms for Image Coding," *Electronics Letters*, vol. 29, pp. 1749-1750, Sept. 1993.
- [32] Hurtgen B. and Hain T. "On the Convergence of Fractal Transforms," *Proc. ICASSP-94*, vol. V, pp. 561-564, 1994.
- [33] Hurtgen B. and Simon S.F. "On the Problem of Convergence in Fractal Coding Schemes," *Proc. ICIP-94*, vol. III, pp. 103-106, 1994.
- [34] Jacobs E.W., Fisher Y., and Boss R.D. "Image Compression: A Study of the Iterated Transform Method," *Signal Processing*, vol. 29, pp. 251-263, Dec. 1992.
- [35] Jacquin A. "A Novel Fractal Block-Coding Technique for Digital Images," in *Proc. ICASSP-90*, pp. 2225-2228, 1990.
- [36] Jacquin A. "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations," *IEEE Trans. Image Processing*, vol. 1, no. 1, pp. 18-30, Jan. 1992.
- [37] Jacquin A. "Fractal Image Coding: A Review," *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1451-1465, Oct. 1993.
- [38] Kawamata M., Nagahisa M., and Higuchi T. "Multi-resolution Tree Search for Iterated Transformation Theory-based Coding," *Proc. ICIP-94*, vol. III, pp. 137-141, 1994.
- [39] Kreyszig E. *Advanced Engineering Mathematics*. John Wiley & Sons, 1993.
- [40] Kominek J. "Understanding Fractal Image Compression," contained in Technical Report CS-95-28, Department of Computer Science, University of Waterloo, pp. 1-44, 1994.
- [41] Kominek J. "Algorithm for Fast Fractal Image Compression," *Proceedings of SPIE: Digital Video Compression: Algorithms and Technologies 1995*, vol. 2419, pp. 296-305, 1995.

- [42] Kumar A., Tannenbaum A.R., and Balas G.J., "Optical Flow: A Curve Evolution Approach", *IEEE Trans. Image Processing*, vol. 5, pp.598-610, April 1996.
- [43] Lazar M.S. and Bruton L.T. "Fractal Block Coding of Digital Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 3, 1994.
- [44] Lepsoy S., Oien G. and Ramstad T. "Attractor Image Compression with a Fast Non-iterative Decoding Algorithm," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, vol. 5, pp. 337-340, Apr. 1993.
- [45] Lin D. "Fractal Image Coding as Generalized Predictive Coding," *Proc. ICIP-94*, vol. III, pp. 117-121, 1994.
- [46] Lu N. *Fractal Imaging*. Academic Press: San Diego, 1997.
- [47] Luo J., Chen C.W., Parker K. J., and Huang T.S., "Artifact Reduction in Low Bit Rate DCT-Based Image Compression", *IEEE Trans. Image Processing*, vol. 5, pp. 1368-1369, Sep 1996.
- [48] Massopust P. *Fractal Functions, Fractal Surfaces, and Wavelets*. Academic Press: San Diego, 1994.
- [49] McGregor D.R., Fryer R.J., Cockshott P., and Murray P. "Faster Fractal Compression," *Dr. Dobb's Journal*, pp. 34-40, Jan. 1996.
- [50] Mokhtarian F. and Mackworth A. "A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 8, pp. 789-805, Aug. 1992.
- [51] Monro D.M. "Class of Fractal Transforms," *Electronics Letters*, vol. 29, no.4, pp. 362-363, 1993.
- [52] Monro D.M. "A Hybrid Fractal Transform," *IEEE Proceedings of ICASSP: Image and Multi-Dimensional Signal Processing*, vol. 5, pp. 169-172, 1993.
- [53] Monro D.M. and Dudbridge F. "Fractal Block Coding of Images," *Electronic Letters*, vol. 28, pp. 1053-1055, May 1992.
- [54] Monro D.M. and Dudbridge F. "Fractal Approximation of Image Blocks," in *Proc. ICASSP-92*, vol. III, pp. 485-488, 1992.
- [55] Monro D.M. and Woolley S.J. "Rate/Distortion in Fractal Compression: Order of Transform and Block Symmetries," *International Symposium on Speech, Image Processing and Neural Networks*, 1994.
- [56] Monro D.M. and Woolley S.J. "Fractal Image Compression Without Searching," *Proceedings of ICASSP*, 1994.

- [57] Naemura T. and Harashima H. "Fractal Coding of a Multi-view 3-D Image," *Proc. ICIP-94*, vol. III, pp. 107-111, 1994.
- [58] Nelson M. *The Data Compression Book*. M&T Publishing: New York, 1992.
- [59] Nicholls J.A. and Monro D.M. "Adaptive Fractal Video," Internal Report, University of Bath, 1995.
- [60] Nitzberg M. and Shiota T., "Nonlinear Image Filtering with Edge and Corner Enhancement," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol 14. no. 8, Aug. 1992.
- [61] Oien G.E. "Parameter Quantization in Fractal Image Coding," *Proc. ICIP-94*, vol. III, pp. 142-146, 1994.
- [62] Oien G.E., Baharav I., Lepsoy S., Karnin E., and Malah D. "A New Improved Collage Theorem with Applications to Multiresolution Fractal Image Coding," *Proc. ICASSP-94*, vol. V, pp. 565-568, 1994.
- [63] Oien G., Lepsoy S., and Ramstad T. "An Inner Product Space Approach to Image Coding by Contractive Transformations," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Toronto, pp. 2773-2776, May 1991.
- [64] Oien G., Lepsoy S., and Ramstad T. "Reducing the Complexity of a Fractal-based Image Coder," *Signal Processing VI: Theories and Applications*, J. Vandevall, R. Boite, M. Moonen and A. Oosterlinck (eds.), Elsevier Science Publishers B.V., pp. 1353-1356, 1992.
- [65] Pei S., Tseng C., and Lin C. "A Parallel Decoding Algorithm for IFS codes Without Transient Behavior," *IEEE Trans. Image Processing*, vol. 5, no. 3, pp. 411-415, Mar.1996.
- [66] Peitgen H., Jurgens H. and Saupe D. *Chaos and Fractals New Frontiers of Science*, Springer-Verlag: New York, 1992.
- [67] Perona P. and Malik J. "Scale-Space and Edge Detection Using Anisotropic Diffusion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629-639, Jul. 1990.
- [68] Sattar F., Floreby L., Salomonsson G., and Lovstrom B. "Image Enhancement Based on a Nonlinear Multiscale Method," *IEEE Trans. Image Processing*, vol. 6, no. 6, pp. 888-895, June 1997.
- [69] Thomas L. and Deravi F., "Pruning of the Transform Space in Block Based Fractal Image Compression," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, vol. 5, pp. 341-344, Apr. 1993.

- [70] Torkamani-Azar R. and Tait K.E. "Image Recovery Using the Anisotropic Diffusion Equation," *IEEE Trans. Image Processing*, vol. 6, no. 5, pp. 1573-1578, Nov. 1996.
- [71] Vaisey J. and Gersho A. "Image Compression with Variable Block Size Segmentation," *IEEE Transactions on Signal Processing*, vol. 40, no. 8, pp. 2040-2060, Aug. 1992
- [72] Van de Walle A. "Relating Fractal Image Compression to Transform Methods," MS Thesis, University of Waterloo, 1995.
- [73] Vines G. and Hayes III M.H. "Adaptive IFS Image Coding with Proximity Maps," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, vol. 5, pp. 341-344, Apr. 1993.
- [74] Woolley S.J. and Monro D.M. "Optimum Parameters for Hybrid Fractal Image Coding," *Proceedings of ICASSP*, 1995.
- [75] You Y., Kaveh M., Xu W., and Tannenbaum A., "Analysis and Design of Anisotropic Diffusion for Image Processing," *Proc. ICIP-94*, pp. 497-501, 1994.
- [76] You Y., Xu W., Tannenbaum A., and Kaveh M. "Behavioral Analysis of Anisotropic Diffusion in Image Processing," *IEEE Trans. Image Processing*, vol. 6, no. 5, pp. 1539-1553, Nov. 1996.

VITA

Nikki McClatchey Bruner

Candidate for the Degree of

Doctor of Philosophy

Thesis: A FAST EDGE PRESERVING FRACTAL SYSTEM

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Denison, Texas, on April 19, 1964, the daughter of George W. and Saroeun H. McClatchey.

Education: Received Bachelor of Science degree and Master of Science degree in Electrical Engineering from Oklahoma State University, Stillwater, Oklahoma in May 1989 and December 1992, respectively; Completed the requirements for the Doctor of Philosophy degree at Oklahoma State University in May, 1998.

Professional Experience: Design Engineer, Device Engineering, from October, 1989 to July, 1990; System Engineer, Electronic Data Systems, from July, 1990 to July, 1991; Research and Teaching Assistant, Oklahoma State University, from August, 1991 to December, 1992, Product Engineer, Seagate Technologies, from December, 1992 to January, 1996, Research Assistant, from November 1994 to Present, Design Engineer, Bruner Consulting, from January 1996 to Present..