To the Graduate Council:

I am submitting herewith a thesis written by Robert William Mauck entitled "The Effects of Limiting the Domain Pool Size on Fractal Image Compression." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Dr. Bruce Whitehead, Major Professor

We have read this thesis
and recommend its acceptance:

D P Neeta

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Robert Mauck

Date August 16, 1993

The Effects of Limiting the Domain Pool Size on Fractal Image Compression

A Thesis

Presented for the Masters of Science

Degree

The University of Tennessee, Knoxville

Robert William Mauck

December, 1993

Dedication

I would like to dedicate this work to my father, Bill Mauck, for his valued advice and for his unwavering support in even the most trying times. I would also like to dedicate this to my mother, Cathy Mauck for her love, and limitless compassion. Without the love and support of both of you I would never have made it this far...

Acknowledgments

I would like to thank my thesis advisor, Dr. Bruce Whitehead, for his advice and support, and direction throughout the development of this research. I would also like to thank Dr. Dinesh Mehta for serving on my thesis committee, as well as Dr. Kimble for his time and advice.

A special thanks goes out to the librarians, Brenda Brooks, Carol Dace, and Mary Lo for their continual quest of references, both here and abroad.

I would especially like to thank Dr. Al Pujol for his encouragement throughout my graduate school career as well as his much needed, and always welcomed advice during the writing of this thesis. This work would not have been possible without the availability of Dr. Pujol's computer facilities.

Abstract

Fractal image compression describes an image by storing the relationship between different regions of an image in the form of contractive affine transforms. This technique provides extremely high compression rates while maintaining good image fidelity and rapid access to the image. The disadvantage of this technique is the intensive computational requirements for image compression. Computational requirements can be reduced by minimizing the number of comparisons that must be made.

A set of domains and ranges were defined over each image compressed. For each range a domain pool was defined consisting of all domains within a given distance (in pixels) from the range. As the size of the domain pool was varied compression ratios and image qualities were measured.

This research concluded that, based on the compression of three images, a distance of approximately 60 pixels resulted in an acceptable compromise between image fidelity and computational requirements. At distances less than 60 pixels increasing the distance results in large benefits in terms of compression ratio and image fidelity. There was little benefit from using distances greater than 60 pixels.

Table of Contents

List of Figures

List of Symbols

SNR(dB)	Signal to Noise Ratio measured in decibels (dB).
e_{ms}	Average Mean Squared Error.
X	Set of all possible images.
u_i	An image from the set X.
u(x,y)	The pixel in the y^{th} row and x^{th} column of the image u as measured from the upper left hand corner.
u*	An approximation to the image u.
u*(x,y)	The pixel in the y^{th} row and x^{th} column of the image u* as measured from the upper left hand corner.
NxM	An image size with N rows and M columns.
PeakDR	The maximum dynamic range of an image.
D	The set of all domains defined over an image.
D_j	The domain j from the set D.
N_D	The number of domains defined over an image.
$\overline{D_i}$	The average intensity of the domain block D_j.
Δd	The distance between the upper left hand corners of the domain blocks.
R	The set of all of the ranges defined over an image
R_i	The range i from the set R.
N_R	The number of ranges defined over an image.
$\overline{R_i}$	The average intensity of the range block R_i.
τ	A collage of transforms which describe an image.
τ_i	The i^{th} transform from the collage of transforms which describe an image.
s	Contractivity of the transform τ.

F_i	The geometric portion of the transform τ_i.
L_i	The Massic portion of the transform τ_i.
c_i	The constant scaling factor for the shade transform τ_i.
α_i	The contrast scale factor for a transform τ_i.
Δg_i	The Brightness scale factor for a transform τ_i.
ι	The set of all the isometric transforms.
ι_{ni}	The isometric n as a part of the transform τ_i.
B	The size of a domain block.
$DP_{(i,T)}$	The domain pool searched to describe the range R_i. Includes all of the domains within T pixels of the range.

Chapter I
Introduction

In today's world of digital computers, graphical images are constantly used as a communication medium for transferring concepts and ideas between computers and users. These images, in the form of picture elements (pixels), are memory and bandwidth intensive. Multiple megabyte hard disk and CD-ROM drives, gigabit magnetic tape drives, and megabit per second (mbps) network communication hardware and software are required to support these images. To increase the capabilities of present computing and communication resources to manipulate and transfer images, the proposed research plans to investigate digital image compression as a method to enhance present computer systems.

Within the past few years, an image compression technique known as fractal image compression was introduced and published in the doctoral dissertation by Jacquin [10]. This dissertation, based on patents held by Barnsley, outlined the algorithms used for this technique [3]. Although this technique provides high image quality and compression ratios, the technique has not been widely accepted because of its intensive computational requirements and extreme compression times.

Fractal image compression techniques encode an image by storing the relationship between different regions of the image to domains defined over the image. The intensive computational requirements of these techniques are a result of the large number of comparisons performed in finding the optimum transform which approximates the image. To reduce the extreme compression times required for compression, the number of comparisons must be minimized. However, reducing the number of comparisons reduces the image quality and compression ratio and increases the memory storage and bandwidth requirements.

For this research, a set of domains and ranges are defined over the image to be compressed. The ranges are the regions of the image that the fractal transforms will describe. The set of domains provide the source of the transforms that will describe the ranges. Determining the transform for a range consists

of comparing the range to a set of domains within a given radius of the range. This set of domains is called the domain pool for that range. The purpose of this thesis is to determine the optimum domain pool radius needed to reduce the computational requirements while maintaining high image fidelities and compression ratios.

Chapter II presents background information and theory of fractal image compression. The chapter introduces image compression techniques, fractals, fractal image compression, the mathematical background of fractals, application of fractal image compression, and theory of effects of varying the domain pool size.

Chapter III presents procedures and algorithms for encoding and decoding an image. Distortion measurements for images, fractal image compression algorithms, fractal image decompression algorithms, and the fractal transform file format are introduced.

Chapter IV presents the results of the research. The results for each compressed image are presented in the form of graphical plots, e.g., compression ratio versus distance, and signal to noise ratio (SNR) versus distance.

Chapter V presents conclusions and areas for future work.

Chapter II
Fractal Background and Theory

This chapter presents background information and the theory of fractal image compression. The chapter introduces image compression techniques, fractals, fractal image compression, the mathematical background of fractals, application of fractal image compression, and the theory of effects of varying the domain pool size. A more detailed discussion of the theory of fractals can be found in Barnsley's text [1].

2.1 Image Compression Techniques

A digital or raster image consists of a two dimensional array of image elements called pixels. Each pixel has an associated intensity or color that is represented by a group of bits. For example, gray scale images have pixel values that range from 0 to 255. Consequently 8 bits must be used to represent each pixel of a gray scale image. True color images are another common raster image format. In this format pixels are made up of three bytes (24 bits), divided up into 1 byte for red intensity, 1 byte for green intensity, and 1 byte for blue intensity. An image is stored in a computer as a sequential stream of pixels with each pixel requiring at least a byte. As the number of pixels in an image increases the storage requirements become very large and can easily exceed a megabyte of disk space. Storage requirements of an image can be reduced by the use of image compression techniques. Image compression reduces the storage requirements of an image by reducing the number of bits per pixel while maintaining the same color resolution. The number of bits of the original file divided by the number of bits of the compressed file is the compression ratio.

There are two families of digital image compression techniques: lossy and lossless. With lossless image compression an image can be compressed and decompressed with no loss of information and the resulting image is identical to the original image. However, these techniques provide relatively low compression ratios. Lossy image compression techniques provide high compression ratios but the

decompressed image is an approximation of the original image. Generally, the higher the compression ratio the lower the image quality.

Over time, standards have evolved in image compression. Two common image compression standards are GIF (Graphics Interchange Format) and JPEG (Joint Photographic Experts Group). JPEG, a collaboration between CCITT (International Telegraph and Telephone Consultative Committee) and ISO (International Standards Organization), is an international standard for compressing and transmitting images[16]. GIF is the standard for transferring images on CompuServe and has gained a wide following in the private and public sectors [5].

GIF is a lossless image compression technique used to store and transmit images. This technique gained its initial popularity on CompuServe, but it has since become a popular standard in networks and bulletin boards in both the public and private sectors [5]. The GIF technique begins with a true color raster image. The raster image is compressed using a slightly modified LZW compression technique [5]. LZW is a lossless compression technique that maps information to a symbol table that is generated dynamically from the data to be compressed. Compression ratios from this technique are typically 2:1 [17].

JPEG is a standard which consists of a group of four separate compression methods. These four forms of JPEG are sequential, progressive, hierarchical, and lossless. In sequential encoding the image is encoded in a single scan [16]. Progressive encoding uses multiple scans increasing the image compression ratio and the compression time [16]. With progressive encoding the image reconstruction proceeds from a coarse initial image to a more refined final image as the scans proceeds. Hierarchical compression encodes the image at multiple resolutions [16]. This way low resolution versions of the image can be accessed quickly and high resolution versions of the image can still be recreated. Lossless encoding provides an exact reproduction of the image after reconstruction [16]. The draw back to this method is that it provides lower compression ratios than the other three lossy techniques. Selection of the method to use is based on properties of the image, computational requirements, and user requirements.

The most common form of JPEG is the sequential compression method [2]. Compression begins by dividing the image into 8 x 8 pixel blocks. A DCT (Discrete Cosine Transform) is applied to each block and the resulting coefficients are stored using Huffman compression. Huffman compression is a lossless technique of representing a piece of information with a variable number of bits. Coefficients that occur frequently are encoded with as few bits as possible while coefficients that occur less frequently will be represented with more bits. Huffman compression requires a table of how the coefficients will be represented. This table is based on probability of occurrence of each coefficient value.

This sequential compression method is lossy due to the truncation error incurred when storing the DCT coefficients. The DCT coefficients are divided by a quantizing value, and the fractional portion of the result is removed. The resulting integers are stored using a Huffman compression. The DCT is reconstructed by decompressing the values and multiplying them by the quantizing factor. The range of integers allowed and the efficiency of the Huffman determine the final compression ratio. If a large range of integers is allowed, DCT coefficients can more accurately be decompressed, but the compressed file requires more storage space. Typically compression ratios of 3:1 or 4:1 can be achieved with sequential JPEG compression while maintaining high quality images[16].

Fractal image compression is a lossy technique that compresses images by storing the self similarities of the image as affine transforms. The transforms can be scaled to operate on images of any size. The affine transforms are size independent. It is difficult to determine the exact compression ratio using this method because the image can be decompressed at a variety of resolutions. Extremely high compression ratios (over 1000:1) have been reported by decompressing a fractal image into an image many times larger than the original [2]. Comparing the compressed image to the original image is a more reliable method of determining the compression ratio. Compression ratios of 10:1 to 40:1 have been reported by comparing the compressed image to the original image file while maintaining high image fidelity. The drawback of this method is that it can take from an hour to days to compress one image on a

80486DX2 66Mhz based computer. However, once the image has been compressed it can be decompressed in a matter of seconds.

Need determines the compression technique used. If the compressed image must be recovered exactly, lossless image compression techniques must be used. The lossless techniques, such as GIF and JPEG lossless, have low compression ratios. If approximation of the original image is acceptable, lossy compression techniques can be used. Lossy techniques, such as JPEG sequential compression and fractal image compression, have high compression ratios. If the computational requirements are not important, fractal image compression may be a good choice. Fractal image compression provides high compression rates and a high image fidelity at the cost of extremely high computational requirements.

2.2 Fractals

The idea of fractals originated with Mandelbrot [9]. In the 1960's Mandelbrot was analyzing cotton prices. Cotton is an old and well regulated commodity and consequently there are well kept records of cotton prices dating back to the early 1900's. After some analysis Mandelbrot noticed a strange peculiarity in the data. Although the day to day prices were random, self similarities on various time scales were emerging. The patterns of one day's prices appeared to be similar to the price trends over a month. After further investigation Mandelbrot confirmed that there were similarities within the data over time and scale. These similarities persisted over a sixty year period that saw two world wars and the American great depression. Mandelbrot had observed fractal behavior in the variation of cotton prices.

Fractals are based on the idea of similarities within a system at various scales. When the cotton data is examined at one scale or another there are no significant similarities. However, when the data at two different scales is examined, e.g. on a monthly and a daily basis, similarities appear. If a mathematical transform can be used to describe the relationship between the information at different scales, the system is called a fractal. The function that describes the relationship is a fractal transform.

Fractal transforms can describe very complex systems with very little information if the system exhibits the fractal characteristics of self similarity at various scales.

After the discovery of similarities in cotton data, Mandelbrot began to explore other occurrences of this fractal phenomenon. Mandelbrot used old mathematical paradoxes to describe and model complex systems and behaviors. He began to see the systems in a whole new light. These old mathematical paradoxes included the two classic fractal examples Cantor dust and Koch curve [9]. These fractal examples can be represented with simple transforms, but they produce complex systems. Mandelbrot's comparisons of these mathematical paradoxes to physical systems resulted in a greater understanding of both the mathematical models and the physical systems.

Cantor Dust is a construction in which a finite line is reduced to an infinite set of points of length zero. This construction begins by taking a finite line and removing the middle third. The resulting two finite line segments are a third as long as the original. Repeating this process on each segment an infinite number of times results in the Cantor Dust construction (Figure 2-1) [1].

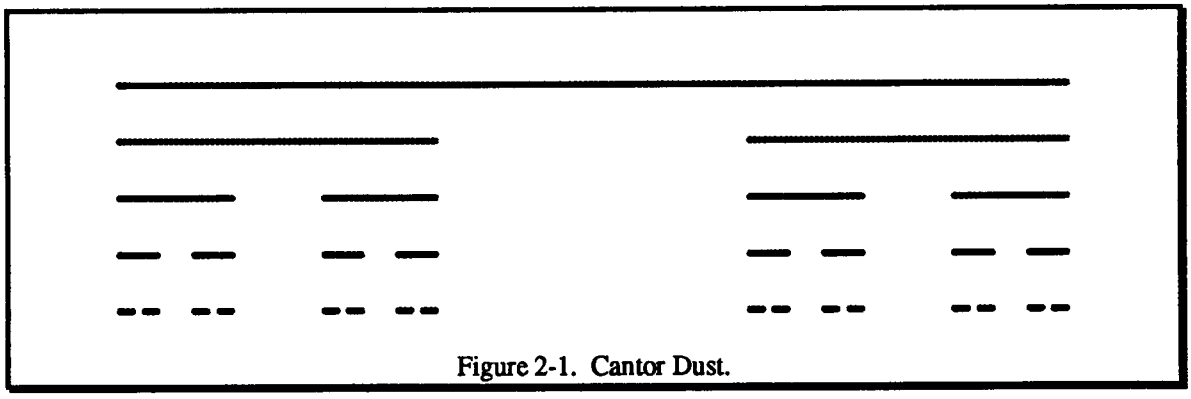


Figure 2-1. Cantor Dust.

This construct was used by Mandelbrot during analysis of noise in telephone lines. In telephone lines, there are periods of noise and periods of no noise. In a period of noise, there were regions of noise and no noise, and so on. This was not very remarkable except for the fact that the ratio of noise to no noise regions remained approximately constant at various scales of observation. From fractal analysis

Mandelbrot concluded that the engineers should stop trying to solve the problem of noise by increasing signal strength. Instead a moderate strength signal should be used and they should accept that there is going to be some error in the signals [9].

A Koch curve is a line segment of infinite length. Construction of a Koch curve begins with a line segment. In the middle of the line segment, make an equilateral triangle with each side a third of the length of the original line segment. Next, remove the middle third of the original line segment. Repeat this procedure on the four line segments an infinite number of times. The resulting curve will be of infinite length (Figure 2-2(a)). A Koch snowflake follows the same procedure as the Koch curve except with an original configuration of three line segments forming an equilateral triangle. The Koch snowflake encloses a finite area, but has an infinite circumference (Figure 2-2(b)).

The Koch curve and Koch snowflake have existed for a long time, but Mandelbrot saw them in a different light. Mandelbrot was contemplating shorelines and political boundaries, and started questioning the accuracy of border measurements. The length of the borders is dependent upon how the measurements are made, just like the length of a Koch segment is dependent upon how many times the procedure has been iterated. When a satellite image is used to measure the length of the border a lot of detail will not be measured and the distance will be approximated in terms of long line segments. The resulting border length will be relatively short. If a yard stick is used to measure length of the border, a longer length will be found because more detail will be taken into account. If a 12 inch ruler is used, the length will be longer again as the measurements go into more of the nooks and crannies. Therefore, the length of the border depends on the scale of the measurement and the detail of the measurement.

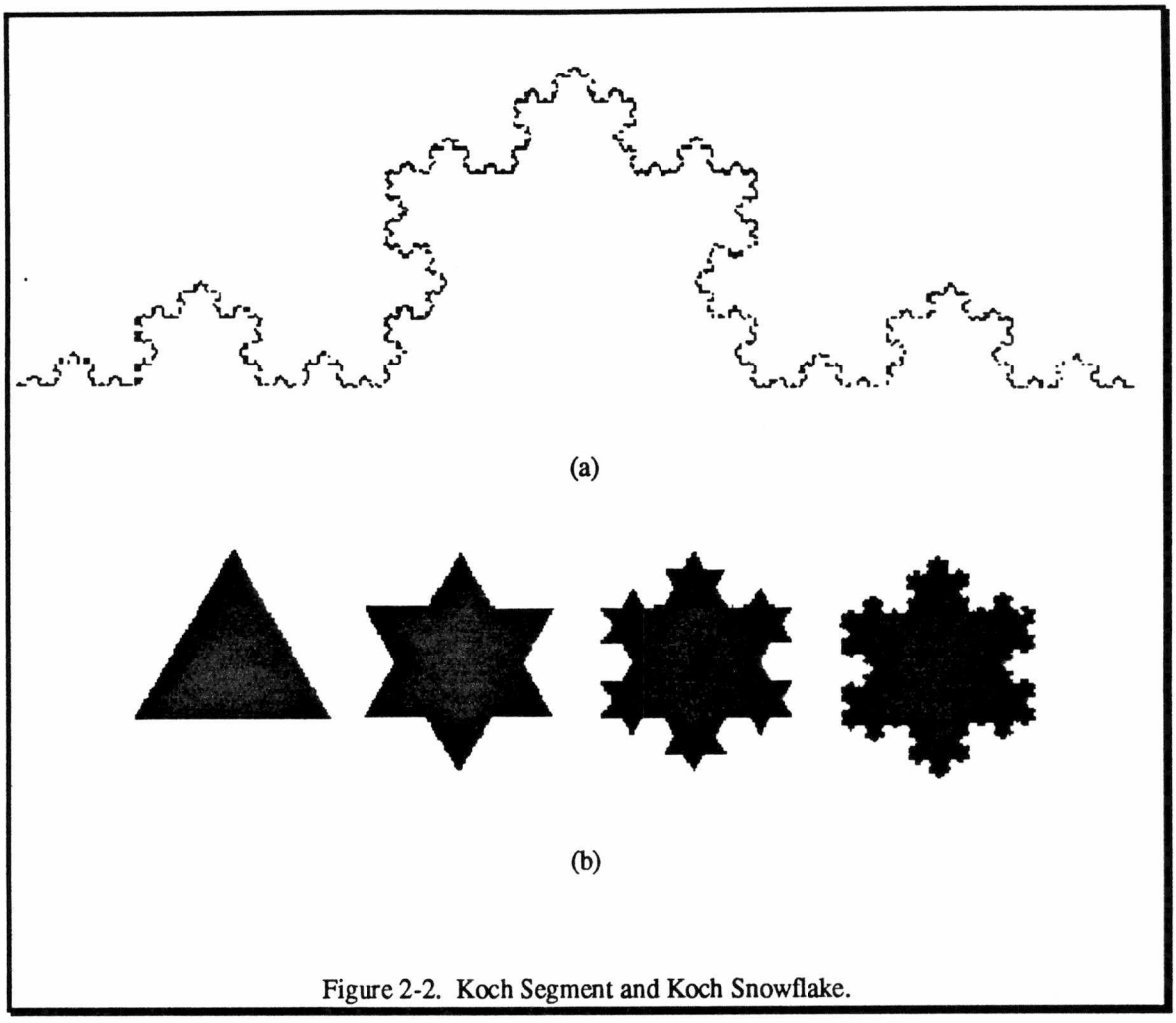


Figure 2-2. Koch Segment and Koch Snowflake.

As the field of fractals and fractal geometry continued to evolve, many interesting systems were explored. Fractal art is a popular area of fractals. Strange mathematical sets, such as the Mandelbrot set, have become very popular for producing images. Fractals can also generated very intricate and colorful images based on simple mathematical formulas that appear to be real world objects such as flowers, trees, and fields [9].

2.3 Fractal Image Compression

If fractals can be used to create pictures, can an arbitrary image be represented with fractals? This is the focus of Barnsley's research. Barnsley is exploring the possibility of representing real world images with fractal transforms. His work has produced very life-like images from relatively simple equations. Initial images were all tediously created by hand but as his work continued Barnsley developed a technique which automates the process of creating a fractal to approximate an arbitrary image.

Since fractals can represent complex systems or images they can be used as an image encoding technique. Because fractals can represent complex structures with very little information images encoded as fractals should take up less storage space and provide an image compression technique. Difficulties lie in finding the fractal transforms that will represent a desired image. The problem of finding the functions is known as the inverse iterative transform problem, and was partially solved by Jacquin [10].

Fractal image compression has large computational requirements due to the large number of transforms considered to find the one transform which best describes the image. Various papers have been published describing various means to compress an image using fractals, but none of these papers have examined how large of a solution set must be searched in order to find an adequate solution which describes the image. Most work in fractal image compression attempts to do some preprocessing to classify regions of the image. Only regions with the same classification are compared [4, 6, 9, 10, 14]. While classifying regions of the image does limit the number of comparisons, the computational requirements are still very large.

Kaouri attempted to limit the size of the solution set by only looking at a small subsection close to the region of the image that was being described [13]. The compression scheme worked. However, no information was presented describing the result of using different sized subsets, or even if different sized subsets were attempted.

2.4 Mathematical Background

This section briefly presents a mathematical framework used to describe iterative systems and resulting fractal images. For a detailed explanation of fractal theory, see Barnsley's books [1, 2].

Let μ denote an image. Each image μ is an element of the set of all possible images X. The distance measure d() is the measure of the similarities between two images. Two images which are visually similar will be a short distance apart, while dissimilar images will have a larger distance. A distance of zero means that the images are identical.

An image, μ, can be operated on by a transform function τ. The transform function, τ, maps the image μ_i to an image μ_{i+1} in the set of all possible images X. s is the contractivity of the transform τ. A transform is contractive if

$$\exists s < 1 \text{ such that } \forall \mu, \nu \in X, d(\tau(\mu), \tau(\nu)) \leq s d(\mu, \nu). \qquad 2\text{-}1$$

When a transform is repeatedly applied to an image it will generate a series of images μ_0, μ_1, μ_2, μ_3 ... μ_n where

$$\{\mu_i = \tau^i(\mu_o)\}_{i \geq 0}. \qquad 2\text{-}2$$

The function τ which does the mapping determines what the series of images will be and how it will behave. If the transform function τ is contractive, the series will converges to an image μ_{orig} regardless of the initial conditions. If $\lim_{n \to \infty} \tau^n(\mu_0) = \mu_{orig}$ then μ_{orig} is the attractor of the transform τ. A contractive transform can be iterated on an image and will converge to a stable image.

By the Collage Theorem, the attractor of a fractal transform can be forced to be arbitrarily close to an initial image, μ_{orig}[1,2,9,14]. This theorem provides the tools needed to solve the inverse transform problem. The Collage Theorem states:

Let $\{\tau : \tau_i, i = 1, ..., n\}$ define a transform function that operates on three dimensional spaces.

If set of transforms τ can be found such that $d(\mu_{orig}, \tau(\mu_{orig})) \leq \varepsilon$, $\varepsilon > 0$.

then

$$d(\mu_{orig}, A) \leq \frac{\varepsilon}{1-s}$$

where A is the attractor of the transform τ, s is the contractivity factor, ε is the maximum allowed error distance[1,14]. Transforms can be made up of many smaller transforms covering the original image. One method of applying this theorem is to cover the desired image with a set of contractive affine transform that map the image onto itself.

2.5 Application of Fractal Image Compression

As previously stated, fractal image compression stores the similarities of an image in the form of affine transforms. The process of determining the affine transforms that represent an image is known as the inverse transform problem. Once the transforms have been determined they are stored in a computer file and used to reconstruct the image. Reconstruction of the image is performed by repeatedly applying the transforms to an initial random image until the image has converged to an approximation of the original image. When transforms are applied to an image, they mutate an image into one the functions represent. The initial image used for reconstruction is unimportant because transforms have an attractor which is the image that they describe. After multiple iterations transform functions will converge on their attractor regardless of initial conditions [1,2,10]. As long as transforms can be stored at a lower bit per pixel rate than the original image, storing fractal transforms can be used as an image compression technique.

Affine transforms are linear functions used in fractal geometry to describe the relationship of one region of an image to another region of the same fractal image. In complex images, groups of affine transforms are used to represent various similarities.

A *three dimensional affine transform* is a linear function described by the function

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & e \end{bmatrix} \bullet \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} + \begin{bmatrix} f \\ g \\ h \end{bmatrix} = \begin{bmatrix} ax_n + by_n + f \\ cx_n + dy_n + h \\ ez_n + h \end{bmatrix} \qquad 2\text{-}4$$

where (x,y) is the location of a pixel. z is the gray scale value of the pixel (x,y). a, b, c and d provide rotation and scaling of the pixels. e scales the gray scale levels. f, g and h provide translation. An affine transform is said to be contractive if it maps a large region onto a smaller region. For the above transform to be contractive a, b, c, d, and e must be greater than or equal to 0 and less than 1. If the transform is contractive, the transform has an attractor[10,11].

As long as affine transforms are contractive, an image can be represented by defining a set of transforms over the entire image known as a collage. The transforms describe how each piece of the original image is similar to another region of the original image. This collage of transforms contains all of the information needed to reconstruct the image in the future [2].

The first step in determining the transforms is to divide the image up into fixed sized, overlapping regions called *domains* D. The set of all domains defined over an image is given by

$$D = \{D_i\}_{0<i\leq N_D} \qquad 2\text{-}5$$

where N_D is the number of domains defined over the image μ. The upper left hand corner of each domain will be offset from the previous domain by a distance of Δd. The domains, D, will provide the domains for the affine transforms.

Next a set of *ranges*, R, are defined. The set of all ranges defined over an image is given by

$$R = \{R_i\}_{0<i\leq N_R} \qquad 2\text{-}6$$

where N_R is the number of ranges defined over the image μ. These non-overlapping ranges cover the entire image μ. Equal sized ranges, half the size of the domains, are initially defined. As the image compression proceeds the size and number of ranges vary.

Domains and ranges cover the entire image. Any given pixel in the image will lie in one range and probably multiple domains.

For each range $R_i \in R$ defined over the image μ a transform τ_i must be found. First the range R_i is classified. Based on the classification of the range R_i a predetermined set of transforms is applied to a subset of the domains D in an attempt to find a match most similar to the range. If the most similar transform-domain pair approximates the range with an error less than the allowed error, a match has been found. If a match is found then the transform that describes that match is stored. Otherwise, the range R_i is split into quarters and another attempt is made at finding a transform for each of the four child ranges. The transforms, the domains they operate on, and the range the transform describe are all stored in a computer file. This file is the compressed image file. The process of defining the subset of the domains D searched for a given transform is described in the next section.

Image decompression begins with an arbitrary image. The arbitrary image is divided up into domains similar to the domains used for compression. Transforms are loaded into memory from the computer file. Next, the ranges are defined over the image based on the information stored in the transforms.

Transforms are applied sequentially to each range defined over the image. The result from the transform replaces the portion of the image covered by the range. One iteration of the fractal transforms is completed when every transform defined over the image has been applied once. After multiple iterations of the fractal transforms an image will converge upon the attractor of the fractal transforms.

2.6 Effects of Varying the Domain Pool Size

This research is an extension of the research by Kaouri [13]. The purpose of this work is to study effects of various sized domain pools on image compression ratio and fidelity. The results create an efficient fractal image compression routine. The number of comparisons required to determine the transforms was minimized while maintaining an acceptable compression ratio and image fidelity.

Let $DP_{(i,T)} \subseteq D$ denote the domain pool used to determine a fractal transform for a given range R_i. The domain pool $DP_{(i,T)}$consists of all domains with upper left hand corners within T pixels of the upper left hand corner of the range R_i (see Figure 2-3). Domain pools are a superset of all domain pools with a smaller distance T. A domain pool includes all smaller domain pools plus new domains for a given range R_i. Increasing the distance T will never cause the image quality or compression ratio to decrease.

The number of computations required is directly proportional to the size of the domain pool. The size of the domain pool increases with the square of the domain pool radius, T^2. The number of domains in the domain pool that must be compared to a range is given by Equation 2-7.

$$NumberOfComparisons \approx \frac{\pi T^2}{\Delta d} \qquad 2\text{-}7$$

Figure 2-4 is the graph for the approximate number of comparisons required if the domains are offset by 4 pixels from each other ($\Delta d=4$). Minimizing the domain pool size while maintaining image fidelity and compression ratio will optimize the computational requirements of fractal image compression.

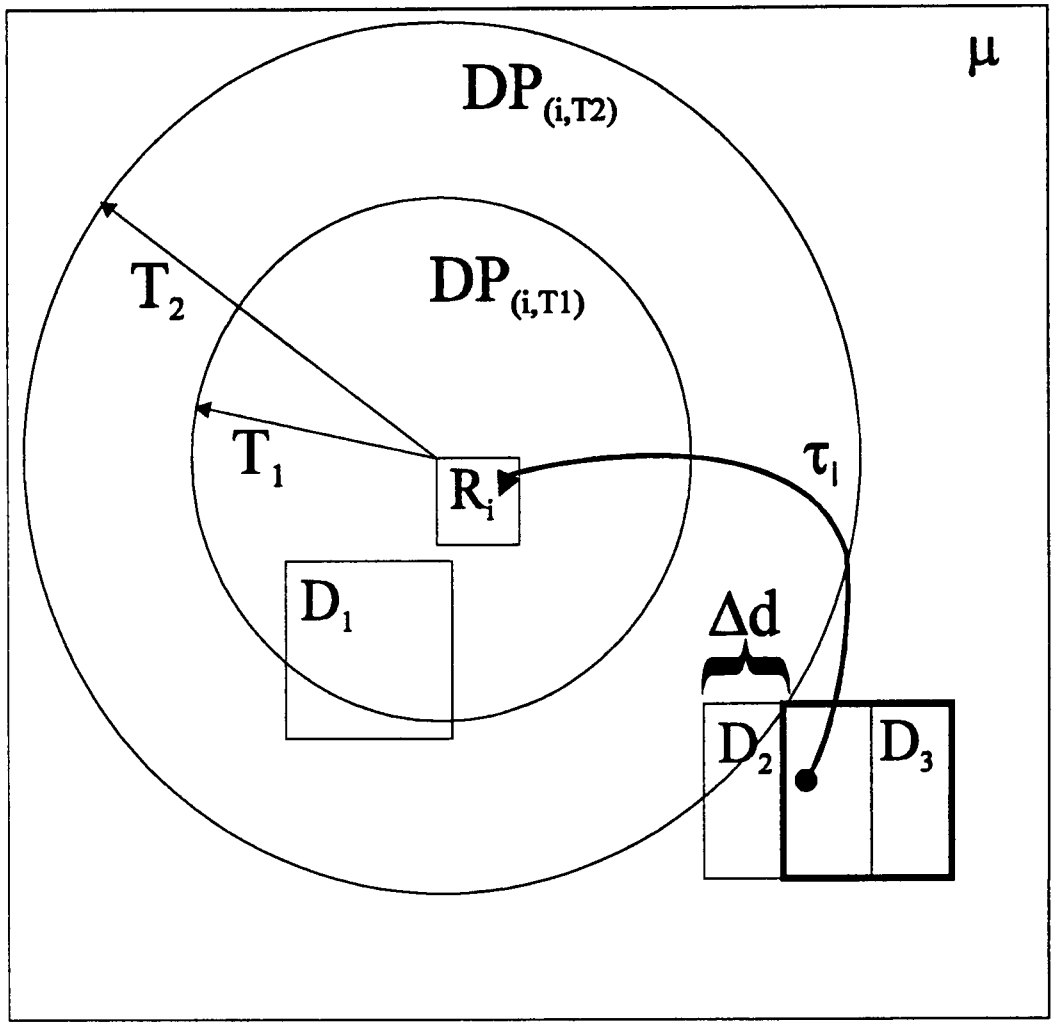


Figure 2-3. Two Domain Pools for the range R_i in the image μ. The domain D_1 is included within the domain pools defined by the radius T_1 and T_2. The domain D_2 is included in the domain pool defined by T_2, but not in the domain pool defined by T_1. τ_i maps the domain D_2 to the range R_i. D_3 is offset from D_2 by Δd.

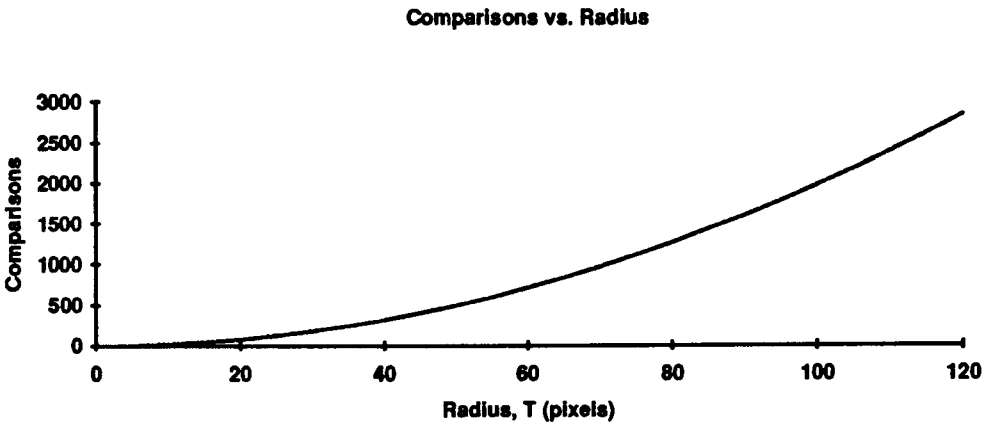


Figure 2-4. Comparisons versus Domain Pool Size Radius T. The domain offset Δd is 4 pixels.

Chapter III
Procedures and Algorithms

This chapter presents procedures and algorithms for encoding and decoding an image. The chapter introduces distortion measurements for images, fractal image compression algorithms, fractal image decompression algorithms, and the fractal transform file format. The basic algorithms covered in this chapter were first presented by Jacquin [10].

3.1 Distortion Measure

Distortion between two images is measured with an average mean squared error [10,12].

$$e_{ms}^2 \cong \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}(u_{i,j}-u_{i,j}^{*})^2 \qquad 3\text{-}1$$

where the image u is N pixels high by M pixels wide. $u_{i,j}$ is a pixel of the current image and $u_{i,j}^{*}$ is the corresponding pixel of expected image. This error provides a measure of similarity between the two images. Although this method does not necessarily measure the visual similarities between images it has been used for many years in the field of image processing and is an accepted measure for image comparison [12]. Other methods of determining the visual similarity between two images exist but they require more computation. The average mean squared method chosen was computationally inexpensive and provided acceptable results [10].

Final analysis of the decompressed image was performed with the signal to noise ratio (SNR) measurement.

$$SNR(dB) = 10\log_{10}\left(\frac{PeakDR^2}{e_{ms}^2}\right)(dB) \qquad 3\text{-}2$$

where PeakDR is the maximum dynamic range of the original image's pixels [12]. In this work the maximum dynamic range of the gray scale images used was 255.

3.2 Fractal Image Compression Algorithms

Encoding an image was accomplished by defining a set of transforms τ. The transforms describe a set of ranges R within a given error. The image to be encoded was initially divided into a set of overlapping domains D of equal size. Next, a range the same size of the original image was defined. The range was then repeatedly quartered until the sub-ranges were smaller than the domains. After the ranges have been defined an exhaustive search was performed to find a transform and a domain describing a range as close as possible. If a domain-transform pair could not be found that described the range to desired accuracy, the range was split into quarters and the process was applied recursively on each quarter. The maximum distortion allowed was determined when the image was encoded.

All of the range sizes as well as the size of the original image to be encoded were assumed to be squares with sides equal to a power of 2. Each pixel was assumed to be an 8 bit value corresponding to one of 256 gray scale intensities.

Each range block was classified as either a shade block, an edge block, or a midrange block. A shade block is defined as a block which has no significant gradient. An edge block has a definite change in intensity within the block. A midrange block has a gentle gradient with no definite edge. For a detailed description of the algorithm used to perform this classification see Appendix A.

The transform describing an image was made up of a collage of transforms. Each transform defines one range of the image. Let $R = \{R_i\}_{0<i\leq N_R}$ define the set ranges defined over an image. An image transform τ has the form:

$$\tau = \sum_{0\leq i<N} \tau_i, \qquad \text{with } \tau_i = \mathrm{F_i} \circ \mathrm{L_i} \tag{3-3}$$

where the affine transform τ_i is made up of two distinct transforms F_i and L_i. The transform τ_i defines the range R_i. Each transform τ_i represents one affine transform as described by Equation 2-4. Actual values for the translation of the affine transform, represented by the variables f and g in Equation 2-4, are

not calculated. Instead, the transform τ_i has a domain index indicating the source for the calculations. The index was used to determine the source domain of the transform. The result represented the range defined by the transform.

L_i is the special contraction of the transform. This transform scaled the domain block to be the same size as the range block. Contraction of the affine transform was performed by this transform. This portion of the transform τ_i corresponds to scaling the variables a, b, c, and d from Equation 2-4.

F_i provided rotation, contrast, and brightness adjustments for the transform. These three parameters were selected to minimize the distortion between the original range and the transformed domain. The contrast α_i and the brightness Δg_i adjustments of the transform correspond to the variables e and h from Equation 2-4 respectively. Rotation of the transform was represented by an isometric transform ι. ι can be represented as a two dimensional rotation matrix that corresponds to the variables a, b, c, and d from Equation 2-4.

Three separate pools of predetermined transforms were searched to determine a transform for a given range. The transform pool searched was determined by the range block classification. The different types of blocks have different characteristics that were exploited to simplify the search for a transform.

If the range was a shade block then the transform used was a constant. The form of the transform was

$$\tau_i = c_i \qquad 3\text{-}4$$

The range block was approximated by a constant value equal to the average intensity of the original range.

If the range was a midrange block then the transform was

$$\tau_i = F_i \circ L_i = \alpha_i (L_i \circ D_j) + \Delta g_i \qquad 3\text{-}5$$

where α_i was a constant scaling factor taken from the set {0.7, 0.8, 0.9, 1.0}, D_j was the domain that the transform was operating on, and Δg_i was calculated such that the average intensities of the original range and the transformed domains were approximately the same. Thus Δg_i was

$$\Delta g_i = \overline{R_i} - \alpha_i \overline{D_j} \qquad 3\text{-}6$$

where $\overline{R_i}$ and $\overline{D_j}$ represent the average intensities if the range and domain respectively.

If the range was classified as an edge block the transform was

$$\tau_i = F_i \circ L_i = \iota_{ni}\left(\alpha_i\left(L_j \circ D_j\right) + \Delta g_i\right) \qquad 3\text{-}7$$

where ι_{ni} was an isometric transform providing rotation and translation of the pixels, α_i was a contrast scale, Δg_i was a brightness scale, and D_j was the domain.

α_i was chosen such that the domain and the range have approximately the same dynamic range. The value for α_i was chosen by calculating

$$\alpha_i = \frac{dr(R_i)}{dr(D_j)} \qquad 3\text{-}8$$

and setting α_i to the closest value from the set {0.5, 0.6, 0.7, 0.8, 0.9, 1.0}. The dynamic range of a block was calculated by identifying the two highest peaks of a histogram of the block. If the peaks were at least 1/4 of the dynamic range apart then the peaks represented the value of the bright and dark regions of the block. If the peaks were not 1/4 of the dynamic range apart, a different method of finding the intensities of the regions was used. To find the value for the light region a search began at the value 0 of the histogram and proceeded upwards until a peak was found. This was an approximate value for the block's light region. The search for the dark region began at the top of the histogram and worked its way down until another peak was found. The second peak represented an approximate value for the block's dark region. The dynamic range of the block was the difference between the values of the peaks.

Δg_i was calculated so the intensity of the dark regions of the scaled domain ($\alpha_i\left(L_j \circ D_j\right)$) and range ($R_i$) were approximately equal. This was done by using the histogram of both regions and then identifying the two highest peaks. Δg_i was found by subtracting the dark peak value of the scaled domain from the dark peak value of the range.

$$\Delta g_i = \text{Dark Peak}(R_i) - \text{Dark Peak}\left(\alpha_i\left(L_j \circ D_j\right)\right). \qquad 3\text{-}9$$

Next the isometric transform was determined. The Isometric transforms rotated and translated pixels within a block. They do not modify pixel values or change the average intensities.

The isometric transform was chosen from a pool of eight possible transforms. Each transform maps a square region to another square region. The eight transforms and their corresponding 2 dimensional matrixes are:

i) Copy the pixels with no reflection or rotation

$$(\iota_0 D)(x,y) = D(x,y) \qquad 3\text{-}10$$

$$\iota_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

ii) Orthogonal reflection about the mid-vertical axis

$$(\iota_1 D)(x,y) = D(B-1-x,y) \qquad 3\text{-}11$$

$$\iota_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

iii) Orthogonal reflection about the mid-horizontal axis

$$(\iota_2 D)(x,y) = D(x,B-1-y) \qquad 3\text{-}12$$

$$\iota_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

iv) Orthogonal reflection about the first diagonal (i=j)

$$(\iota_3 D)(x,y) = D(y,x) \qquad 3\text{-}13$$

$$\iota_3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

v) Orthogonal reflection about the second diagonal (i + j = B - 1)

$$(\iota_4 D)(x,y) = D(B-1-y,B-1-x) \qquad 3\text{-}14$$

$$\iota_4 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

vi) Rotation about the center of the block +90°

$$(\iota_5 D)(x,y) = D(y, B-1-x) \qquad 3\text{-}15$$

$$\iota_5 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

vii) Rotation about the center of the block +180°

$$(\iota_6 D)(x,y) = D(B-1-x, B-1-y) \qquad 3\text{-}16$$

$$\iota_6 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

viii) Rotation about the center of the block -90°

$$(\iota_7 D)(x,y) = D(B-1-x, y) \qquad 3\text{-}17$$

$$\iota_7 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Each domain block D is of size BxB. Each isometric transform was tried and compared to the original range. The isometric transform ι_k producing the lowest distortion was chosen for the transform τ_i that describes the given range R_i.

For each range, the transform was applied to a set of domains D_R where $D_j \in D_R$. The domain D_j and the parameters from transform $\tau_i(D_j)$ that resulted with the lowest distortion compared to the original range R_i was the transform chosen. If the distortion was larger than a predefined acceptable amount, the range was quartered. The procedure was applied recursively to each of the quartered ranges.

The domain pool used to determine a transform for a range R_i was the set of all domains within the circle of radius T_d centered on the upper left hand corner of the range. For a domain to be included in the circle the upper left hand corner of the domain must lie in the circle. Domains that lie outside the radius T_d are not considered as candidates for the given range and ignored.

Transforms were determined for each range R_i defined over the image μ. As each transform was determined it was placed in the fractal transform file. Details of the file format are in the section Fractal Transform File Format.

3.3 Fractal Image Decompression Algorithms

Decompression begins by defining an image μ with arbitrary pixel values. The initial μ image consisted of uniform gray scale values of 128. The transforms are loaded into memory from the compressed file. Ranges and domains are then defined over the image based on the transforms from the file. The transforms are then iterated over the image. As the transforms were iterated the image converged towards the attractor of the transforms. The fidelity of the final decompressed image depends on the number of times the transforms were iterated on the image and the error bounds used to determine the transforms.

For this work the transforms were applied 10 times to an initially uniform gray image with pixel values of 128. Previous work done by Jacquin has determined that a transform will converge to the desired image with ten iterations of the transforms [10].

3.4 Fractal Transform File Format

The transforms describing an image were stored in a computer file. The compression ratio was the ratio of the transform computer file to the original file size. An efficient means of storing the transforms maximized the compression ratio. This program used a variable record storage scheme dependent on the transform type and range block size.

The first 21 bits of the fractal transform file consisted of a file header containing the size of the original image, size of the domains, and offset of one domain to another domain. The header is detailed in Table 3-1. All three of these fields are unsigned integers.

Table 3-1. IFS File Header Format.

Information	Original Size	Domain Size	Domain Shift
Bits	9	8	4

Following the header were descriptions of the affine transforms. Each transform was described by a variable length record which follow each other sequentially. The record format of each entry is detailed in Table 3-2. Each record consists of a minimum of a Transform, Δg_i sign bit, a value for Δg_i, and the size flag.

The transform field was a 3 bit unsigned integer indicating the transform type of this record. It corresponded to one of the three classification types: 0 for shade transform, 1 for midrange transform, and 2 for an edge transform.

The meaning of Δg_i varied for each transform, but every transform had an 8 bit value for Δg_i and a sign bit for Δg_i.

Table 3-2. IFS File Record Format.

Information	Transform	Δg_i Sign	Δg_i	Size Flag	$\log_2$ Size	Domain	α	isometric
Bits	2	1	8	1	3	0/12/12	0/2/3	0/0/3

The size flag indicated if the record represented a new size. If the size flag was set then the next 3 bits represented the size of the range this record and the next three records represented (which may recursively split into smaller regions themselves). These three bits were the $\log_2$ of the size of the range. The actual range size was determined by taking 2 to the power of this value.

If the record represented a midrange or an edge transform, the next 12 bits indicated the domain index used for this transform. The domains are indexed from left to right, starting at the top and moving down the image.

If the record represented a midrange block, a value for α_i was in the next 2 bits. The actual value for α_i was calculated by taking the value stored in the two bits, adding 7 to it and multiplying by 0.1. The result was a number in the set {0.7, 0.8, 0.9, 1.0}.

If the record was an edge block, fields which followed the domain are 3 bits for α_i and 3 bits for the isometric transform ι_i. α_i was calculated by adding 2 to the value for α_i and multiplying by 0.1. This resulted in a value for α_i from the set {0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 0.9, 1.0}. The bits from the isometric field correspond to one of the eight isometric transforms.

Although this variable file format was moderately complicated it provided an efficient usage of bits in the transforms file. The longest one record could be was 33 bits long. The size field was only used in 1/4 of the records. The maximum length of records without the length field was 25 bits. A group of four transforms took 108 bits. Gray scale blocks 2 pixels wide and 2 pixels high block contained 128 bits of raster information. As long as range blocks were 2x2 or larger the transforms provided compression. Since the size of the transforms was fixed larger range regions produced higher compression ratios. The smallest range block permitted was 4x4 pixels. Therefore, this algorithm guaranteed image compression.

Chapter IV
Results

This chapter presents the results of the research. Results for each compressed image are presented in the form of graphical plots, e.g., compression ratio versus radius, and signal to noise ratio (SNR) versus radius.

Figures 4-1(a), 4-3(a), and 4-5(a) illustrate three 256x256 gray scale images that were compressed using the fractal image compression technique. Each image was initialized with a range size of 256x256 pixels and a domain size of 32x32 pixels. The range for each image was repeatedly quartered until the three dimensional affine transform (Equation 2-4) approximated the range within an average mean square error of 1.0, or the range reached a size of 4x4 pixels. The average mean squared error of 1.0 provided a SNR (Equation 3-2) of approximately 48.13dB. Referring to Section 2.5, the overlapping parameter, Δd, for each domain was set to 4 pixels, i.e., the position of the upper left hand corner of each domain was offset from the previous domain by 4 pixels.

Referring to Section 2.6, each image was compressed using multiple domain pool sizes. For each domain pool size, the number of bytes representing the compressed image was recorded and compression ratios determined by dividing the original image size by the compressed image size.

The SNR was determined by comparing the decompressed image for each domain pool size to the original image. Referring to Section 3.3, each image was decompressed by iterating the fractal transforms 10 times. The transforms were applied to a uniform gray image with a pixel value of 128. SNR and Compression Ratios versus Domain Pool Radius, defined as the domain pool size (Section 2.6), are illustrated in Figures 4-2, 4-4, and 4-6.

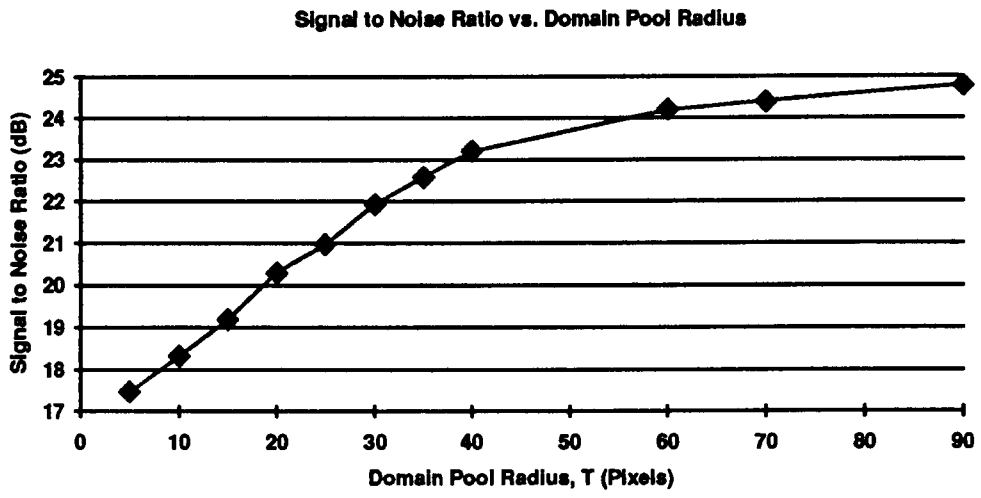
(a) (b)

(c) (d)

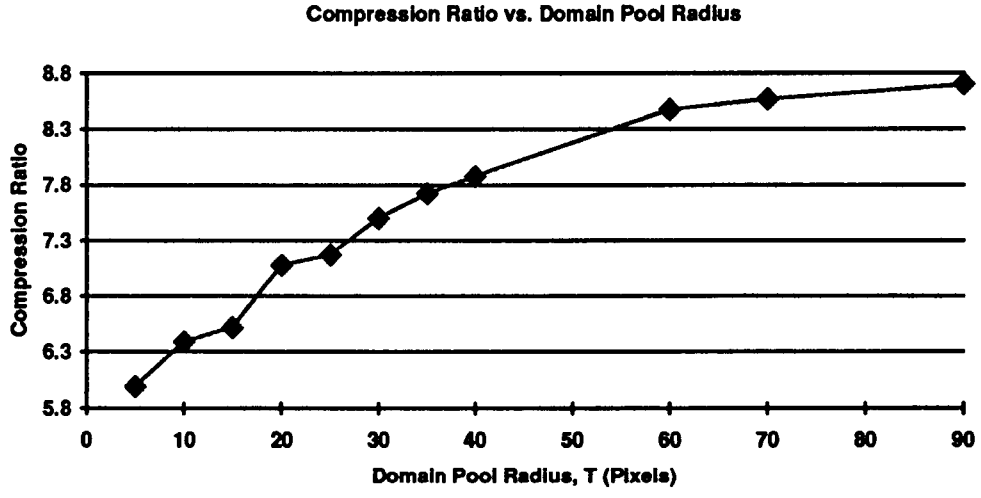
Figure 4-1. LENA Original and Decompressed Images.

(a) The original Image. (b) Compressed with T=20 pixels.

(c) Compressed with T=60 pixels. (d) Compressed with T=90 pixels.

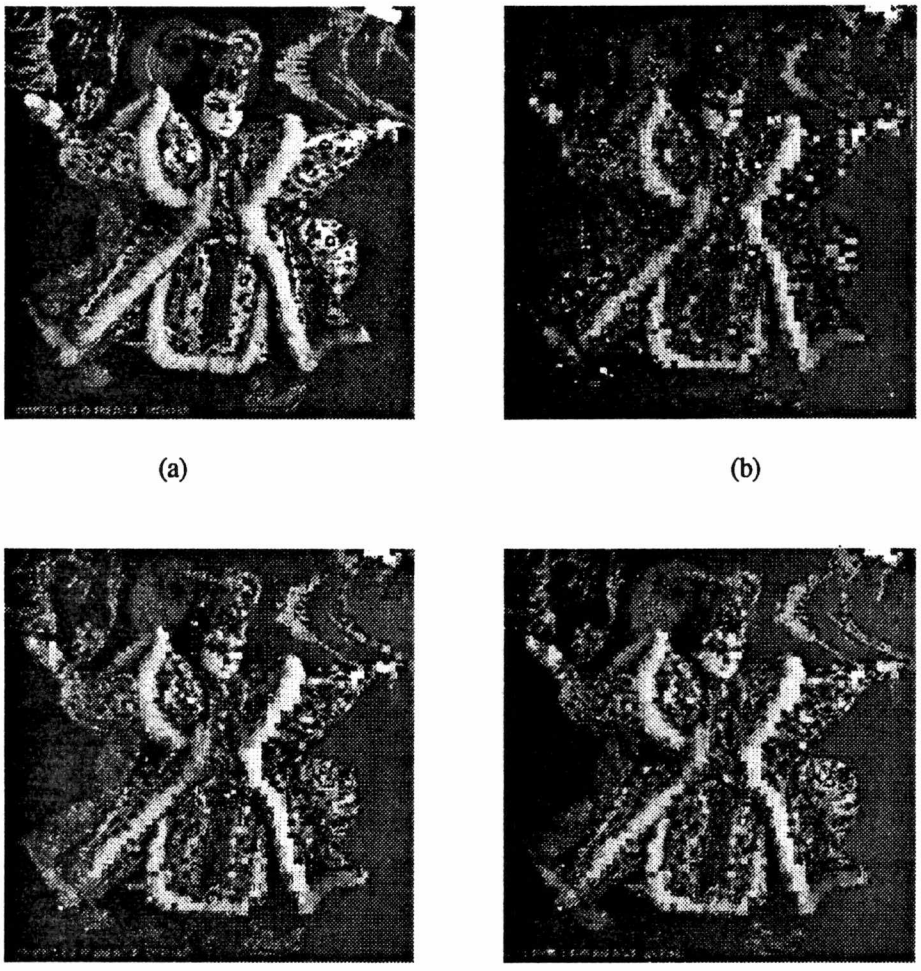


(a)



(b)

Figure 4-2. Compression results for LENA.



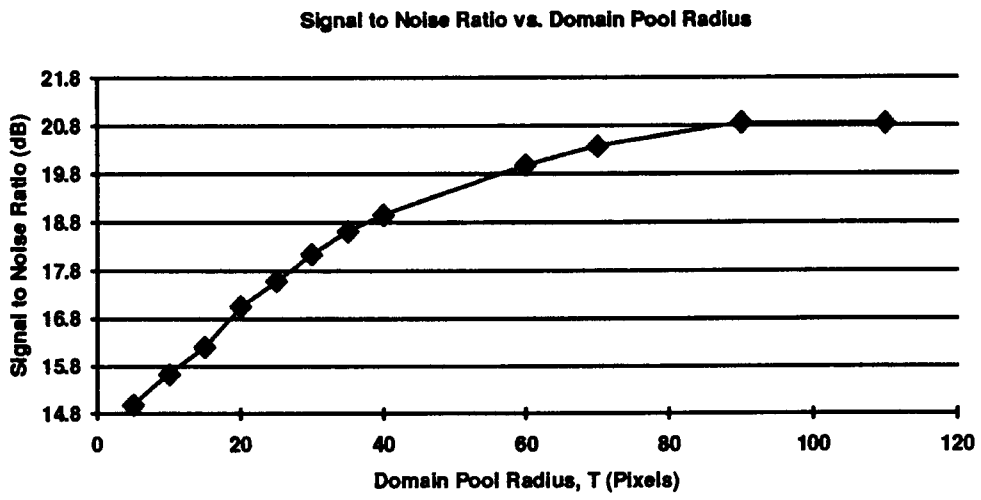
(a) (b)

(c) (d)

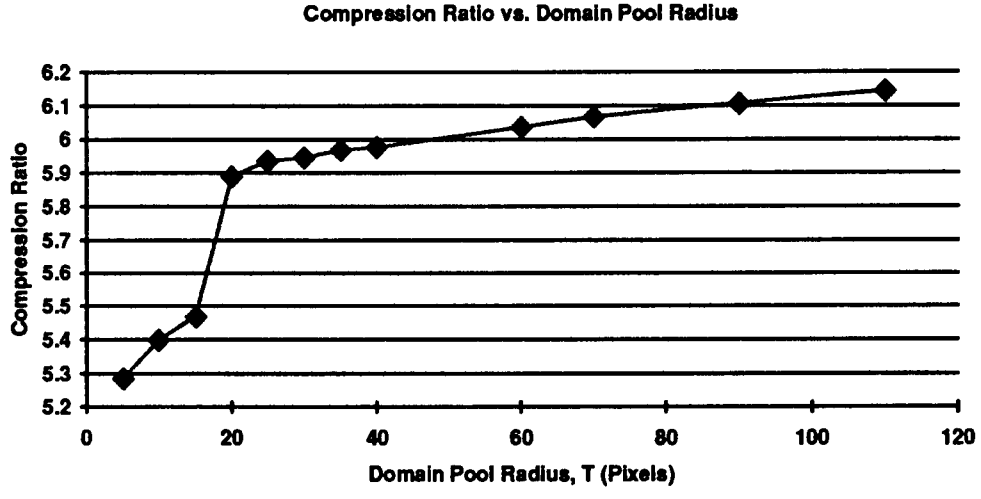
Figure 4-3. Doll Original and Decompressed Images.

(a) The original Image. (b) Compressed with T=20 pixels.

(c) Compressed with T=60 pixels. (d) Compressed with T=90 pixels.

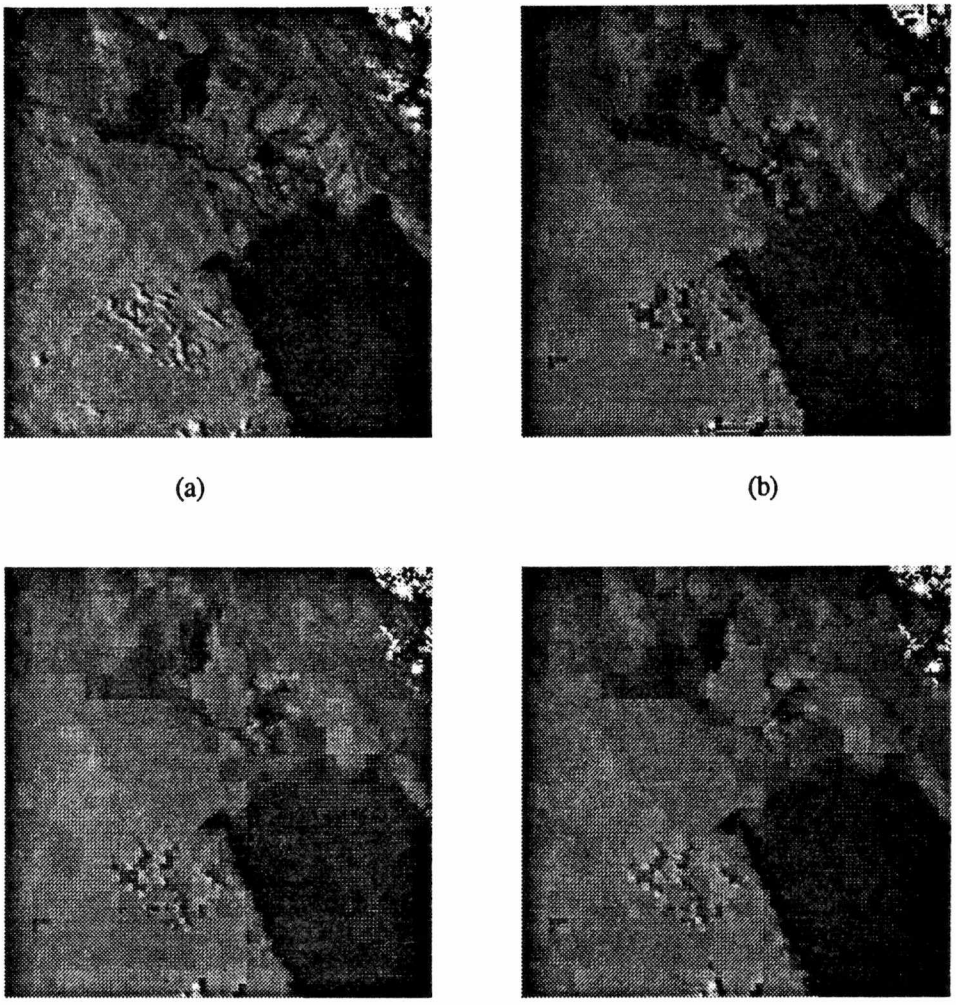


(a)



(b)

Figure 4-4. Compression results for DOLL.



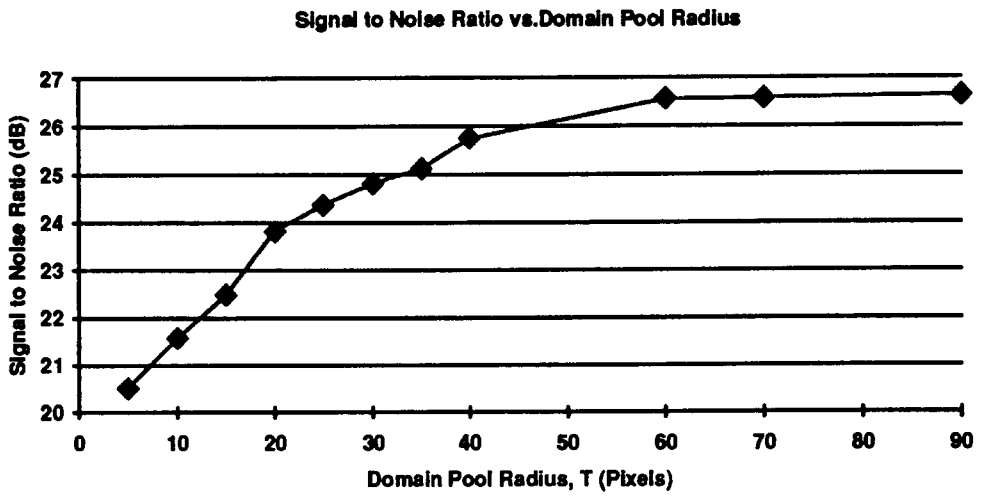
(a) (b)

(c) (d)

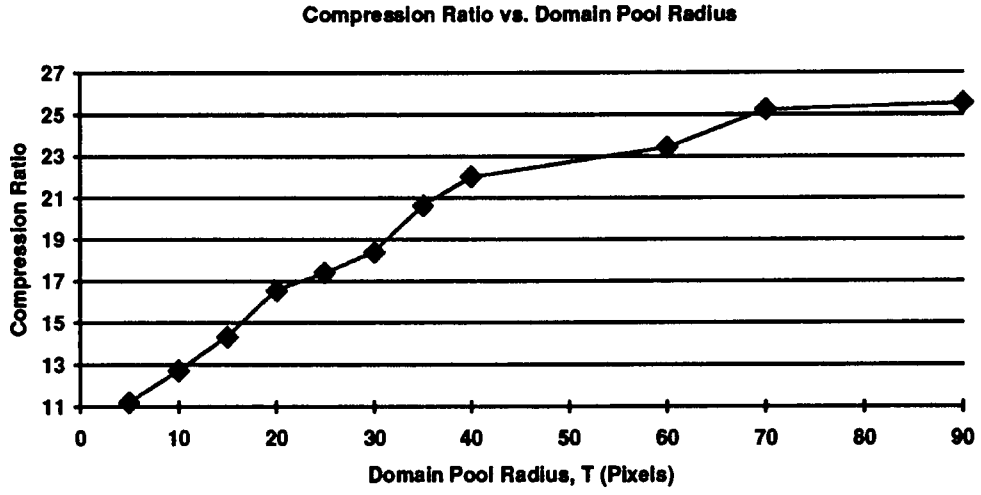
Figure 4-5. GULF Original and Decompressed Images.

(a) The original Image. (b) Compressed with T=20 pixels.

(c) Compressed with T=60 pixels. (d) Compressed with T=90 pixels.



(a)



(b)

Figure 4-6. Compression results for GULF.

4.1 LENA Results

Figure 4-1(a) illustrates the original working image LENA. This image was compressed 11 times using domain pool radius, T, set to values {5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 60.0, 70.0, 90.0}. Figure 4-1(b), (c) and (d) illustrate three additional images using T=20, T=60, and T=90, respectively. The image fidelity (measured by the SNR) and compression ratio are shown in Figure 4-2(a) and (b) respectively.

Referring to Figure 4-1, as T increases the images illustrate a better approximation to the original image. Specifically, for T=20 pixels (Figure 4-1(b)) the compressed image was very blocky. Image resolution or quality was jagged and somewhat undefined. Finer details, such as individual feathers attached to the lady's hat, were indistinguishable. Other details within the image scene had the same characteristics. A radius of T=60 provided a better approximation of the original image (Figure 4-1(c)). The quality became more definite and individual feathers in the lady's hat began to appear. Additional improvement in image quality are seen for T=90 (Figure 4-1(d)). Also, image quality increased more rapidly for T=20 to T=60, than for T=60 to T=90. This contention is confirmed by visual observations of Figure 4-2(a). The figure shows that the SNR ratio increases rapidly (from 17.46bB to 23.2dB) for increasing domain pool radius (from 5 to 40), and goes into saturation for increasing radius T.

The compression ratio versus radius is shown in Figure 4-2(b). As the radius T increases compression ratio increases. Compression ratio does not improve as rapidly at higher radii.

Referring to figure 4-2(b), the compression ratio increases rapidly (from xx:1 to xx:1) for increasing domain pool radius (from 5 to 40), and also goes into saturation for increasing radius T. This contention also confirms the above observations.

4.2 DOLL Results

Figure 4-3(a) shows the original image of a doll. This image was compressed 12 times with T equal to {5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 60.0, 70.0, 90.0, 110.0}. The three compressed image are shown in Figure 4-3(b), (c), and (d). The image fidelity and compression ratio results are shown in Figure 4-4(a) and (b) respectively.

Referring to Figure 4-3, as T increases the images illustrate better approximations to the original image. This result is confirmed by Figure 4-4(a). When the domain pool radius increases beyond T=70 the SNR goes into saturation. These results are similar to the results from the image LENA.

Referring to figure 4-4(b), the compression ratio increases rapidly (from xx:1 to xx:1) for increasing domain pool radius (from 5 to 30), and also goes into saturation for increasing radius T. These results are also similar to the results from the image LENA.

4.3 GULF Results

Figure 4-5(a) shows the original image of GULF. This image was compressed 11 times with T equal to {5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 60.0, 70.0, 90.0}. The three compressed image are shown in Figure 4-5(b), (c), and (d). The image fidelity and compression ratio results are shown in Figure 4-6(a) and (b) respectively.

Referring to Figure 4-5, as T increases the images illustrate better approximations to the original image. This result is confirmed by Figure 4-6(a). When the domain pool radius increases beyond T=60 the SNR goes into saturation. These results are similar to the results from the images LENA and DOLL.

Referring to figure 4-6(b), the compression ratio increases rapidly (from xx:1 to xx:1) for increasing domain pool radius (from 5 to 60), and also goes into saturation for increasing radius T. These results are also similar to the results from the image LENA and DOLL.

4.4 Overall Results

In all three cases image compression ratio and image fidelity increased with the domain pool radius T. However, both compression ratio and image fidelity became saturated as T increased beyond 60 pixels. The number of comparisons also increases rapidly with T (see Figure 2-3). Consequently, compressing images with radii greater than 60 to 80 pixels is computationally expensive and does not increased image fidelity or compression ratio. This result is consistent for both image fidelity and compression ratio of all three images.

Chapter V
Conclusions

A fractal image compression algorithm was developed to minimize the computational requirements for compressing images. The algorithm encoded images by storing relationships between different regions in the image to domains defined over the image. Three images were compressed with various domain pool radii T. Image fidelities and compression ratios were recorded and analyzed for each compressed image. Domain pool radii less than 60 pixels resulted in low image qualities and low compression ratios. Domain pool radii greater than 60 pixels provided little improvement in image quality or compression ratio. Also, the computational requirements to compress the image increased with the square of the radius of the domain pool, T. Therefore, this research concludes that a radius T of 60 pixels provided acceptable compression ratios, image fidelities, and minimal computational requirements. Further research should modify the transforms to increase image quality and determine the effects of various domain pool sizes with the optimized transforms.

List of References

List of References

[1] Barnsley, Michael F. *Fractals Everywhere*. Academic Press, INC. San Diego, CA. 1988.

[2] Barnsley, Michael F. and Hurd, Lyman P. *Fractal Image Compression*. AK Peters, Ltd. Wellesley, MA. 1993.

[3] Barnsley, and Sloan, United States Patent # 5065447, "Method and Apparatus for Image Compression." 1991.

[4] Beaumont, J. M. "Image Data Compression Using Fractal Techniques." BT Technology Journal. Vol 9 No 4 October 1991.

[5] CompuServe. "GIF Graphics Interchange Format, A Standard Defining a Mechanism for the Storage and Transmission of Raster-Based Graphics Information." CompuServe, Incorporated. June 15, 1987.

[6] Dettmer, Roger. "Form and Functions - Fractal-Based Image Compression." IEEE Review, September 1992.

[7] Fisher, Y., Jacobs, E. W., and Boss, R. D. "Fractal Image Compression Using Iterated Transforms." *Image and Text Compression*. Storer, J. A. Kluwer Academic Publishers, Dordrecht, Netherlands. 1992.

[8] Fisher, Yuval. "Fractal Image Compression - SIGGRAPH '92 Course Notes." The San Diego Super Computer Center, University of California, San Diego CA. 1992.

[9] Gleick, James. *Chaos, Making a New Science*. Penguin Books, New York, New York. 1987.

[10] Jacquin, Arnaud E., Ph.D. "A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding." U.M.I. Ann Arbor, MI. 1989.

[11] Jacquin, Arnaud E. "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations." IEEE Transactions on Image Processing, Vol 1, No 1, January 1992.

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

[13] Kaouri, H. A. "Fractal Coding of Still Images" International Conference on Digital Processing of Signals in Communications: 6th: 2-6 September 1991: University of Lomughborough, U.K. London: Institute of Electrical Engineers c1991.

[14] Kocsis, S. M. "Fractal-Based Image Compression" 23rd Asilomar Conference on Signals, Systems & Computers. 1989. Pacific Grove, California.

[15] Ramamurthi, Bhaskar and Gersho, Allen. "Classified Vector Quantization of Images" IEEE Transactions on Communications, Vol. COM-34, No. 11, November 1986.

[16] Wallace, Gregory K. "The JPEG Still Picture Compression Standard." Communications of the ACM, Vol 34, No 4, April 1991.

[17] Welch, Terry A. "A Technique for High-Performance Data Compression." IEEE Computer, Vol 17, No 6, June 1984.

Appendix

Block Classification

This appendix describes the algorithm used to classify a block into either an edge, a midrange, or a shade block. This algorithm is based on the work by Ramamurthi and Gersho and also by Jacquin [15, 9, 10]. A modified version of the algorithm is presented here.

Classification of the block is a two step process. The first step is to enhance and analyze the image which results in two gradient tables G_h and G_v and some counters S_h, S_v, H_p, H_n, V_p, and V_n. Next the counters are examined and a decision tree is used to determine the block type. The result is one of three cases. The block is either an edge block, a midrange block, or a shade block. A shade block is a block with a no discernible gradient. A midrange block is a block with a smooth gradient over the block but without a definite boarder between a light and dark region. An edge block is one with a definite gradient and a definite boarder between the light and dark regions.

Let u(i0, j0, B) denote an image block of size BxB with the upper left hand corner at (i0, j0). Let $u_{i,j}$ denote the gray scale level at the point i,j. Let u_{avg} be the average intensity of the block u.

Begin by calculating the threshold values Te, Ts, Js and Je. The edge contrast threshold Te and the shade contrast threshold are define as

$$T_e = \begin{cases} \dfrac{8.0}{\mu_{avg}} \text{ if } \mu_{avg} < 30.0 \\ 0.2 \text{ otherwise} \end{cases} \qquad \text{A-1}$$

and

$$T_s = \begin{cases} 0.1 \text{ if } \mu_{avg} < 30.0 \text{ or } \mu_{avg} > 225.0 \\ 0.025 \text{ otherwise} \end{cases} \qquad \text{A-2}$$

The thresholds Js and Je which are used to filter out false results are defined as

$$J_s = B - 1 \qquad \text{A-3}$$

and

$$J_e = \frac{B}{2.0}. \qquad \text{A-4}$$

The elements of the normalized horizontal and vertical gradient tables are given by

$$d_h(i,j) = \frac{2(\mu_{i,j} - \mu_{i+1,j})}{\mu_{i,j} + \mu_{i+1,j}} \qquad \text{A-5}$$

and

$$d_v(i,j) = \frac{2(\mu_{i,j} - \mu_{i,j+1})}{\mu_{i,j} + \mu_{i,j+1}}. \qquad \text{A-6}$$

For the rest of this section assume that d_h and d_v are equivalent to $d_h(i,j)$ and $d_v(i,j)$ respectively.

The gradient tables are generated by comparing d_h and d_v to an edge contrast threshold T_e. Whenever the absolute value of the normalized gradient is greater than the shade contrast threshold T_s then the shade counters S_h and S_v are incremented.

A-7

$$G_h(i,j) = \begin{cases} 1 \text{ if } d_h > T_e \\ -1 \text{ if } d_h < -T_e \\ 0 \text{ otherwise} \end{cases}, \text{ with } i_0 \le i \le i_0 + B - 2 \text{ and } j_0 \le j \le j_0 + B - 1$$

and

$$S_h += 1 \text{ if } |d_h| > T_s. \qquad \text{A-8}$$

Likewise

A-9

$$G_v(i,j) = \begin{cases} 1 \text{ if } d_v > T_e \\ -1 \text{ if } d_v < -T_e \\ 0 \text{ otherwise} \end{cases}, \text{ with } i_0 \le i \le i_0 + B - 1 \text{ and } j_0 \le j \le j_0 + B - 2$$

and

$$S_v += 1 \text{ if } |d_v| > T_s. \qquad \text{A-10}$$

Now the polarity of the gradient is examined. A positive and negative polarity, corresponding to the black to white or white to black gradient of the block, is determined for both the horizontal and

vertical gradient tables. The values are stored in H_p and H_n for positive and negative polarity from the horizontal table. V_p and V_n store the results for positive and negative polarity from the vertical table. The value H_p is the number of +1's from the table G_h. H_n is the count of -1's from the table G_h. Likewise V_p and V_n correspond to the number of +1's and -1's respectively from the vertical gradient table G_v.

The block can now be classified. If $Vp \geq Je$ or $Vn \geq Je$ or $Hp \geq Je$ or $Hn \geq Je$ then the block is classified as an edge block. If the block is not and edge block then if $Sh \leq Js$ and $Sv \leq Js$ then the block is a shade block. If the block does not pass either of the previous tests then the block is a midrange block.

VITA

Robert William Mauck, son of Cathy and Major Buford (Bill) Mauck, was born in Harrisonburg, Virginia on June 29, 1969. He graduated from Oakton High School in June, 1987. The following August he enrolled in Bridgewater College. In May of 1991 he received a Bachelor of Science for Math/Computer Science and Physics. In August, 1991 he entered the University of Tennessee Space Institute and in December of 1993 received a Masters of Science in Computer Science.

Robert is now attempting to live life to its fullest, avoiding stress and trying to have a good time. Thesis recovery may take 9-10 years of rest and relaxation. After his recovery he plans to get a job at Disney World as either an Imagineer, or as one of those people who goes around all day selling balloons.