5-1-1995

Implementation of fractal image coding

Peter Stubler

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Title: Implementation of Fractal Image Coding

I, Peter O. Stubler, hereby grant permission to the Wallace Memorial Library of Rochester Institute of Technology to reproduce my graduate project in whole or part.

Signature: _______________________

Date: 5/25/95
Abstract

The goal of this project is to implement a digital image encoder and decoder using a Fractal Block Coding compression algorithm for grayscale images, and to compare its performance to currently popular algorithms such as JPEG. The algorithm used here is based on the published papers [1] - [3] of A. E. Jacquin, and in part, a paper [4] by B. Ramamurthi and A. Gersho. As stated in the project proposal, this algorithm has been simplified to enable the timely completion of the project.
# Table of Contents

Graduate Project Release Permission Form.................................................i
Abstract ........................................................................................................ii
Table of Contents .......................................................................................iii
List of Figures ...............................................................................................iv
List of Tables .................................................................................................iv
List of Plates ................................................................................................iv
Glossary of Terms ........................................................................................v
Introduction ..................................................................................................1
About Contractive Operations .....................................................................1
Description of the Fractal Block Encoding ..............................................2
  Range and Domain Blocks .......................................................................3
  Parent and Child Blocks .........................................................................5
  Transformations .......................................................................................5
  Geometric Part (Si) ...................................................................................6
  Massic Part (Ti) .......................................................................................6
  Contractivities .........................................................................................7
Measures of Distortion and PSNR .............................................................8
Description of the Encoding Algorithm ..................................................8
  Image Block Classification ......................................................................9
  Pools of Domain Blocks .........................................................................9
  Encoding the Image Partition ................................................................10
  Quantization of α and Δg .......................................................................11
Description of the Decoding Algorithm ................................................12
Discussion of Results ................................................................................14
  Block Classes ..........................................................................................14
  Encoding Parameter Statistics ...............................................................14
  Iterative Decoding ..................................................................................20
  Block Artifacts .......................................................................................21
  Contractivities Revisited .......................................................................21
  PSNR and Subjective Image Quality ......................................................24
  Compression Ratios ...............................................................................26
  Other Results ..........................................................................................27
Conclusion ................................................................................................27
References ..................................................................................................29
Appendix A: Source Code ..........................................................................30
List of Figures
Figure 1: Iterations of Contractive Expressions ........................................ 2
Figure 2: Calculation of Resultant Range Block via Transformation of Domain Block ........................................ 4
Figure 3: Isometric Transformations on Domain Blocks ................................ 6
Figure 4: Statistics on Transformations for Lynn ........................................ 15
Figure 5: Statistics on Transformations for Lena ....................................... 16
Figure 6: Statistics on Transformations for Pyramid .................................. 17
Figure 7: Statistics on Transformations for Jennifer ................................... 18

List of Tables
Table 1: Contractivities of Transformations ............................................. 7
Table 2: Bit Allocation for Block Codes by Class ....................................... 11
Table 3: PSNR of Intermediate Images in Decoding Process ....................... 19
Table 4: Average Contractivity by Block Class and Partition ....................... 22
Table 5: Block Class Distribution by Partition for Pyramid ......................... 23
Table 6: Comparison of PSNR (dB), Compression Ratio, and Subjective Image Quality ........................................ 25

List of Plates
Plate 1: Partial Encoding Techniques ..................................................... P1
Plate 2: Iterations of Fractal Image Decoder for Lynn ................................ P2
Plate 3: Iterations of Fractal Image Decoder for Lena ................................ P3
Plate 4: Iterations of Fractal Image Decoder for Jennifer .......................... P4
Plate 5: Transmography Pyramid into Lena ............................................. P5
Plate 6: Fractal Reconstructed Image vs. Original (Lynn) ......................... P6
Plate 7: Fractal Reconstructed Image vs. Original (Lena) .......................... P7
Plate 8: Fractal Reconstructed Image vs. Original (Jennifer) ..................... P8
Plate 9: Fractal Reconstructed Image vs. Original (Pyramid) .................... P9
Plate 10: Divergent Fractal Image vs. Original (Pyramid) ......................... P10
Plate 11: Iterations of an Unstable Fractal Code ..................................... P11
Glossary of Terms

absorption page 6
A transformation on an image block which sets the luminance of each pixel in the block to the same specified level.

cchild block page 5
A small \((nxn)\) image block. Four child blocks comprise a \((2nx2n)\) parent block.

contractivity page 1
The measure of how quickly an expression, when iterated, will converge to a predeterminate value.

DCT page 25
Discrete Cosine Transform. A mathematical transformation similar to Fourier Transform which produces values having no imaginary component. Basis for JPEG compression standard.

domain block page 3
A \(2mx2m\) image block whose pixel values will be transformed to create a resultant range block \((mxm)\) for use elsewhere in the reconstructed image.

dynamic range page 10
The range in luminance values contained within an image block.

geometric transformation page 6
A 2:1 spatial contraction performed on every domain block in the process of creating a resultant range block.

JPEG compression page 24
Joint Photographic Experts Group compression standard based on quantized Discrete Cosine Transforms of image blocks.

lossy image compression page 1
Any compression mechanism which introduces distortion into the recovered (decompressed) image.

massic transformation page 6
One of several operations performed on a domain block in the process of creating a resultant range block. These include contrast scaling, luminance shifting, and reorientation.

original range block page 4
The part of the original image currently being encoded. The encoding process searches for a domain block and
transformation set which will produce a resultant range block closest to the original range block.

**parent block**
A large \((2n \times 2n)\) image block. Four child \((n \times n)\) blocks comprise a parent block.

**PSNR**
Peak Signal to Noise Ratio. A measure of image distortion used in this paper.

**resultant range block**
The part of the image currently being decoded, produced by executing the transformations prescribed by encoding process. Also found in the encoding process when the encoder calculates the resultant image block in order to measure the distortion in the search for the optimum domain block/transformation combination.

**vector quantization (VQ)**
A lossy image compression mechanism which quantizes an image block as the member of a codebook of image blocks which results in the lowest measured distortion.
Introduction

Fractal images are synthetic images which have a high degree of visual complexity, yet low information content. That is because they are generated by recursively applying a relatively simple algorithm in such a way that any region of the resultant image can be represented as a transformation of another region of the same image.

The “lossy” image encoding algorithm described by Jacquin is predicated on two assumptions: 1) most images contain redundant information; and 2) this redundancy can be exploited by describing one section of an image as a transformed version of some other section of the same image. These transforming operations include absorption, scaling, rotating, mirroring, contrast scaling, and luminance shifting. The key to the algorithm is choosing operations which are contractive in nature and which will, when iterated, converge to a steady state. In this case, the desired steady state is an approximation of the original image.

About Contractive Operations

Consider the following expression: $9(x)^{1/2}$. Given any positive, non-zero value for $x$, evaluate the expression and store the resultant value in $x$. As this process is iterated, the value of $x$ will converge to $81$. The expression $9(x)^{1/2}$ is a contractive operation whose resultant value is an approximation of $81$. Suppose the storage needed to record the expression $9(x)^{1/2}$ is less than the storage needed to record the value $81$. If so, one could save data storage and/or transmission time by storing the expression $9(x)^{1/2}$ anywhere the value $81$ is needed. The costs involved in doing this are 1) the time required to calculate the
approximation of the value 81, and 2) the inaccuracy of the resultant value due to the approximation process.

Consider another expression: \(27(x)^{1/4}\). This expression also converges to 81, but given the same initial value of \(x\), does so more quickly than the first expression. The second expression is said to be more contractive than the first. Figure 1 illustrates iterated evaluations of these expressions with an initial value of \(x\) equal to 1.0.

The contractivity of an expression is often represented as a positive scalar value \(s\). As \(s\) approaches zero, the expression converges more quickly towards its value limit. If \(s\) equals 1, the expression ceases to be contractive, and at values greater than one the expression becomes expansive.

![Figure 1: Iterations of Contractive Expressions](image)

**Description of the Fractal Block Encoding**

Like many image compression algorithms, Jacquin's algorithm encodes images in a blockwise fashion. The image is segmented into
non-overlapping, rectilinearly aligned square blocks. Each of these blocks are encoded such that the average number of bits to encode each block is less than the number of bits in the original block.

Recall that this encoding algorithm is predicated on the assumption that any section of an image can be represented as a transformation of another section of the same image. Thus, to encode a section of an image, we search through the image looking for another section of the image which when manipulated (transformed) by one or more contractive operators produces the best approximation of the original. Of course, the section itself transformed by a unity operator would be the best choice indicated by such a search, but because the unity operator is non-contractive, the block itself is explicitly excluded from the search. The bits encoding each block represent a contractive expression comprised of one or more affine transformations which when iterated will produce an approximation of the original image block.

Conceptually, this algorithm has many similarities to Vector Quantization (VQ). While VQ uses a codebook agreed upon a priori which must be transmitted, Fractal Block Encoding uses the original image as a “virtual codebook” which aides in the specification of the affine image transformations which form the contractive expressions. The “virtual codebook” is not transmitted, only the contractive expressions.

RANGE AND DOMAIN BLOCKS

As stated above, we encode one section of an image as a transformed version of some other section of the same image. To support this, Jacquin introduces the concept of range blocks and
domain blocks. For clarity, we will extend these terms to include original range block and resultant range block. The term original range block refers to the section of the image being encoded. The term domain block refers to the section of the image which will be transformed to create an approximation of the original range block called a resultant range block. The original image can be seen as a collection of original range blocks while the decoded image can be seen as a collection of resultant range blocks which are each approximations of their corresponding original range blocks. The fractal code for a range block may specify a domain block and the transformation(s) used to create the resultant range block. Stated another way, we use domain blocks as the “raw material” for creating resultant range blocks.

Figure 2: Calculation of Resultant Range Block via Transformation of Domain Block
PARENT AND CHILD BLOCKS

Jacquin's algorithm uses two block sizes, parent blocks measuring eight pixels square (8x8), and child blocks measuring four pixels square (4x4). Each parent block is comprised of four child blocks. The image is first partitioned and encoded as a collection of 8x8 parent blocks. For each parent block in the encoded image, the distortion between each of its component child resultant range blocks and the corresponding child original range blocks is measured. If the distortion for a given child block is not below a target threshold, the child block is independently encoded with the results of the child block encoding superseding (covering) that portion of the parent block. If more than two child blocks need to be independently encoded, the fractal block code for the parent block is discarded, and the region is simply encoded as a collection of four child blocks.

As put forth in the original project proposal, for this paper, we restrict our attention to encoding images using only small (4x4) child blocks. This simplifies the algorithm which allows this project to be completed in a reasonable time frame. This simplification results in a decrease in the compression ratios obtained because we cannot take advantage of encoding large regions at the parent level. It also may increase the encoding time for the same reason. It should not, however, degrade the resultant image quality since we are encoding at a more detailed level.

TRANSFORMATIONS

The image block transformations can be broken down into two parts: the Geometric Part (S_i) and the Massic Part (T_i) and can be represented as follows:
Range Block ≈ Ti (Si (Domain Block))

In the process of transforming the domain block into a range block the geometric transformation is applied first, followed by the massic transformation(s).

GEOMETRIC PART (Si)

The geometric operator (S) is a spatial contractor which maps domain blocks to range blocks. In this implementation, domain blocks are defined to be twice the size of their corresponding range blocks. For example, if the range block is 4x4 a corresponding domain block must be 8x8. If the range block is 8x8 the corresponding domain block must be 16x16. Thus, our geometric operator scales down the domain block by setting each range block pixel to the average value of the group of four pixels at the corresponding location in the domain block. In [3], Jacquin shows that this operator has a contractivity of 1.

MASSIC PART (Ti)

The massic transformations applied to a scaled down domain block may include one or more of the following operations:

i) Absorption at gray level g0, where \(0 \leq g0 \leq 255\). This operation simply sets the value of all the pixels in the block to some uniform gray level g0.

ii) Contrast scaling by \(\alpha\). This operation multiplies the value of each pixel in the block by \(\alpha\) with the resultant value clipped to the range of 0 to 255. Values of \(\alpha\) less than 1 will decrease the dynamic range of the resultant block, while values of \(\alpha\) greater than 1 will increase the dynamic range of the resultant block.

iii) Luminance shift by \(\Delta g\), where \(-255 \leq \Delta g \leq 255\). This operation shifts the luminance of the block by adding \(\Delta g\) to the value of
every pixel in the block. Note that again, the resulting value for each pixel is limited to the range of 0 to 255.

iv) One of eight isometric transformations illustrated below in Figure 3. These transformations simply shuffle the pixels in a deterministic way, and thus effectively increases the pool of domain blocks by a factor of eight.

<table>
<thead>
<tr>
<th>Identity</th>
<th>Reflection about mid-vertical axis</th>
<th>Reflection about mid-horizontal axis</th>
<th>Reflection about first diagonal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reflection about second diagonal</td>
<td>Rotate +90°</td>
<td>Rotate +180°</td>
</tr>
</tbody>
</table>

Figure 3: Isometric Transformations on Domain Blocks

**CONTRACTIVITIES**

The contractivity of all of the transformations are listed in the Table 1 below. Note that only absorption at $g_0$ and contrast scaling by $\alpha$ where $\alpha < 1$ are contractive in nature. The other transformations are non-expansive, and when $\alpha > 1$ contrast scaling can become expansive. In Jacquin's original work, he limited the range of $\alpha$ from 0 to 1 to ensure convergence to a single point approximation of the original image. In [1] he references work by Jacobs and Fisher [5] which indicates that a value of $\alpha_{\text{max}}$ equal to 1.5 is allowable, and will result in convergence to a stable decoded image. This point will be discussed in the results section.
<table>
<thead>
<tr>
<th>Transformation</th>
<th>Contractivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial Contraction</td>
<td>$s = 1$</td>
</tr>
<tr>
<td>Absorption at $g_0$</td>
<td>$s = 0$</td>
</tr>
<tr>
<td>Luminance shift by $\Delta g$</td>
<td>$s = 1$</td>
</tr>
<tr>
<td>Contrast scaling by $\alpha$</td>
<td>$s = \alpha^2$</td>
</tr>
<tr>
<td>Isometries</td>
<td>$s = 1$</td>
</tr>
</tbody>
</table>

Table 1: Contractivities of Transformations

**Measures of Distortion and PSNR**

Although the human eye is the ultimate discriminator of image quality for pictorial data, we must find acceptable alternatives when dealing with image processing algorithms. For image block comparison, the measure of distortion is defined as the sum over the image block of the square of the difference between corresponding pixels. For image comparison, we calculate a peak signal-to-noise ratio (PSNR) over the entire image using the same method as documented in [5]. This value is calculated as follows:

$$\text{PSNR} = -20 \log_{10} \left( \frac{\text{rms}}{2^n - 1} \right),$$

where $n$ is the number of bits per pixel.

**Description of the Encoding Algorithm**

The goal of the encoding algorithm is to find for each $nxn$ region of the image (original range block) another region of the same image measuring $2nx2n$ (domain block) which when manipulated with a combination of the above mentioned transforms will result in a good approximation of the original (resultant range block). The most appropriate domain block indicated by the search is called a *matching block*. An exhaustive search of the image for a matching block is possible, but would be prohibitively time consuming. Instead, the
image is partitioned into partitions measuring 256x256 pixels which are encoded independently, and image block classification is used to further narrow the breadth of the search for matching blocks within each partition.

**IMAGE BLOCK CLASSIFICATION**

The goal of image block classification is to decrease the search time for the matching block, and increase the quality of the resultant image by discarding a priori a large percentage of potential candidates. Several methods of block classification have been devised. The method employed by Jacquin is one borrowed from Vector Quantization.

In [4], Ramamurthi and Gersho proposes a technique to improve the performance of Vector Quantization by preserving edge information within image blocks. The technique entails encoding each image block based on its geometric visual characteristics. Prior to encoding, each block is classified as one of several classes. Only "codebook" vectors of the same class are used in encoding the image block. The major classifications are: *shade blocks*, *midrange blocks*, and *edge blocks*. A shade block has an almost uniform gray level with little or no perceptible gradient. A midrange block has a mild gradient but no strong edges. An edge block contains one or more strong changes in intensity and often contains part of the boundary of an object in the image. We use the same image block classifier to determine how a range block will be encoded and which domain blocks are reasonable candidates for a match.

**POOLS OF DOMAIN BLOCKS**

Each partition (256x256) of the image is encoded independently. The partition is first contracted spatially 2:1 into a
domain image. Pools of domain blocks are generated by passing an $n \times n$ window over the domain image in horizontal and vertical increments of $n/2$, and classifying each block. Blocks classified as shade blocks are ignored, but tables are built for midrange and edge blocks. The midrange domain block table contains the address of each midrange block within the domain image and its mean gray level and dynamic range. Similarly, the edge block table contains the same information for all edge blocks. These tables comprise our pools of domain blocks.

ENCODING THE IMAGE PARTITION

The image partition is segmented into original range blocks by passing an $n \times n$ window over it in horizontal and vertical increments of $n$ such that each pixel within the partition is included in exactly one range block. Each range block is encoded according to its class.

Shade blocks are simply encoded as a uniform gray level set to the mean gray level of the block quantized to six bits. (Absorption at $g_0$.) No domain block is referenced or needed (which explains why no table was built).

Midrange blocks are encoded by applying a contrast scale factor and a luminance shift to a midrange domain block. For each block recorded in the midrange domain block table, we calculate the required contrast scale factor and luminance shift to best approximate the range block. The mean gray level and dynamic range stored in the tables are used to calculate contrast scale factors and luminance shift values. These values are calculated as follows:

$$\alpha = \frac{\text{dynamic range (range block)}}{\text{dynamic range (domain block)}}$$

$$\Delta g = \text{mean(range block)} - \alpha \text{ mean(domain block)}$$
We then calculate the sum-squared-error between the transformed domain block and the range block. The domain block which yields the smallest sum-squared-error is the one used in the encoding process. The encoding information includes the location within the partition of the domain block, the value of $\alpha$ quantized to 5 bits, and the value of $\Delta g$ quantized to 7 bits.

Edge blocks are encoded in much the same way as midrange blocks except the sum-squared-error calculation is repeated for each of eight reorientations (isometries). The best fitting isometry is encoded in 3 bits.

Limiting $\alpha_{max}$ to 1.0 ensures convergence of the resultant image but comes at the cost of image fidelity. To duplicate a range block's dynamic range, we seek to find a matching domain block with a dynamic range at least as large. If $\alpha_{max}$ is allowed to grow larger than 1.0, we have a better opportunity to replicate the range block's dynamic range.

<table>
<thead>
<tr>
<th>Shade Blocks</th>
<th>Midrange Blocks</th>
<th>Edge Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Description</td>
<td>Bits</td>
</tr>
<tr>
<td>2</td>
<td>block type</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Gray Level</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>domain X</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>$\alpha$ contrast scale</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>$\Delta g$ gray offset</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>orientation</td>
<td>8</td>
</tr>
</tbody>
</table>

8 Total | 26 Total | 29 Total

Table 2: Bit Allocation for Block Codes by Class.

**Quantization of $\alpha$ and $\Delta g$**

In [1], Jacquin quantized values for $\alpha$ to one of four values less than or equal to one. These values differed for parent and child blocks, but each was encoded in two bits. In [5], Jacobs et al. conducted a thorough study to determine optimal number of bits for the
quantization of $\alpha$ and $\Delta g$. The study concluded that the values were 5 and 7 respectively.

It is important to note that Jacobs et al. allowed $\alpha$ to range from $-\alpha_{\max}$ to $+\alpha_{\max}$, and quantized $\alpha$ logarithmically. Jacobs notes that there appears to be a higher density of $\alpha$ values near $\alpha$ and $\Delta g$. In this implementation, we do not need a sign bit, but use five bits to quantize values of $\alpha$ linearly in a range of 0.03125 to 2.0 and choose a value of $\alpha_{\max}$. This was done for ease of implementation, flexibility in specification of $\alpha_{\max}$, and speed in the calculation of normalized pixel values (by substituting a shift of 4 rather than a divide by 16). This is not particularly efficient bit usage, but if $\alpha_{\max}$ was equal to 1.0, we would have both speed and efficiency by eliminating the fifth bit altogether.

**Description of the Decoding Algorithm**

The result of the encoding process is series of contractive expressions which will, when iterated, produce an approximation of the original image. While the encoding process is simply a search for the best fitting contractive expressions, the decoding process is analogous to the approximation of the value 81 in the example above. In the example, we began by choosing an initial value for $x$. For our image decoding process, we initialize our image plane to some value. It is important to note that the initial contents of the image plane is not significant. The image plane could be initialized to a flat field of all white, black, gray, or some unrelated image having the same dimensions.

In each iteration of the decoding process, the decoder evaluates the expression for each resultant range block in the image, and
measures the change in the image from the previous iteration. The change in the image between iterations resembles an exponential decay curve. The decoding process stops when the change between iterations approaches zero which indicates that the image has reached a steady state.

The expressions evaluated by the decoder are simply the transforms specified by the encoding process. These expressions are encoded into a stream of bits. For each block, the first two bits specify the block class. For shade blocks, the next 6 bits represent a quantization of the gray level with which to fill the block. For midrange and edge blocks, continue with the following values: $y$ position of the domain block within the partition (6 bits), $x$ position of the domain block within the partition (6 bits), $\alpha$ (5 bits), and $\Delta g$ (7 bits). Edge blocks finish with 3 bits which specify the isometric transformation (reorientation) of the domain block. Remember that for both midrange and edge blocks, the domain block is first scaled down by a factor of two.

Because a domain block is twice the size of its corresponding range block, it contains four blocks which were range blocks in the previous iteration of the decoding process. Consider the following: A midrange range block will refer to a midrange domain block. That midrange domain block may specify a region of the image composed of four shade range blocks. Each of these four shade blocks will attain their steady state value after the first iteration. In this case, the midrange range block will then attain its steady state value after the second iteration. Although this example is somewhat contrived, we can see that the geometric (spatial contraction) transform, while non-
contractive itself, helps distribute some contractivity of the other transfroms, and speeds convergence.

**Discussion of Results**

**BLOCK CLASSES**

Plate 1 illustrates the results obtained by using different sets of block class encoding mechanisms. Encoding all blocks as shade blocks results in severe image degradation especially in blocks containing a significant gradient. Encoding blocks as either shade or midrange blocks significantly improves the reconstructed image, but also significantly increases the encoding and decoding times. The increase in encoding time is due to the search of all midrange domain blocks for each midrange range block. The increase in decoding time is due to the need to iterate the decoding process for midrange (and edge) blocks. Further improvements in image quality can be obtained by encoding blocks as either shade, midrange, or edge blocks. This results in an increase in encoding time, but has no significant impact on decoding time. The increase in encoding time is due to the search of every reorientation of every domain edge block for each edge range block. Because midrange blocks and edge blocks are decoded essentially the same way, the decoding time is roughly the same.

**ENCODING PARAMETER STATISTICS**

Figures 4, 5, 6, and 7 illustrate statistics drawn from the fractal codes for Lynn, Lena, Jennifer, and Pyramid respectively. The distribution of blocks into classes appears to be, as expected, image dependent. The distribution for range blocks and domain blocks is roughly constant for an image.
Figure 4: Statistics on Transformations For Lynn
Figure 5: Statistics on Transformations for Lena
Figure 6: Statistics on Transformations for Pyramid
Figure 7: Statistics on Transformations for Jennifer
**Domain to Range Distances:** The domain to range block graph has the same shape for all images. It seems that the probability that a domain block is an optimal match for a range block is inversely proportional to the distance between them. The possibility arises to take advantage of this characteristic to increase compression and/or run-time performance. Entropy encoding the location of the domain block relative to the range block may decrease in the average bit rate for domain block addresses. Ordering the table of candidate blocks, or searching for matching domain blocks in such a way as to begin in the immediate vicinity of the range block, may decrease search time. This would, of course, require a target threshold to terminate the search early, possibly leading to a less than optimal encoding.

**Contrast Scale Factors:** The spike in the contrast scale factor at $\alpha_{\text{max}}$ is another opportunity to save some bits. This occurs simply due to the clipping of $\alpha$ to $\alpha_{\text{max}}$ when it exceeds the limit. Below $\alpha_{\text{max}}$ there does not seem to be enough consistency to warrant any further entropy encoding.

**Luminance Offsets:** The luminance offset histogram peaks at zero but is asymmetric about that point. This is due to the asymmetrical range of $\alpha$ about 1.0. More $\alpha$ values below 1.0 require more $\Delta g$ values above 0 to compensate for them. Note that the range of $\Delta g$ is symmetric around 0. An improvement may be to make the range of $\Delta g$ asymmetric to balance the distribution. Additionally, entropy encoding may be considered to decrease the average bit rate.

**Isometries:** Each image uses a different distribution of isometric transformations, but there are some similarities between them. The identity transform is the most frequently used in each of the images.
The least frequently used are the ±90° rotations. Although not consistently second in frequency the 180° rotation is often used.

**ITERATIVE DECODING**

Plates 2, 3, 4, and 5 illustrate the iterative nature of the decoding process. Before the decoding process begins, the image frame is initialized to some value. The initial contents of the image frame is not significant. For convenience the image frame has been initialized to the value 0x80 in Plates 2, 3, and 4. In Plate 5 we initialize the image to that of a Mayan Pyramid. We can see after the first iteration that the shade class blocks have already reached their steady state values. In the following iterations we can see the contents of the midrange and edge blocks evolve from uniform groups of pixels 4x4 to 2x2 to single pixel groups. This effect reflects the activity of the Geometric transform (scaling).

It is easy to see that these images are encoded and decoded as a collection of four independent partitions each measuring 256 pixels square. Partitions with a low aggregate contractivity values \( s \) reach a steady state quicker than those partitions with high values of \( s \). The PSNR values for the first eight iterations, the final result, and the compression ratio are shown in Table 3.

<table>
<thead>
<tr>
<th>Name</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>Final</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lynn‡</td>
<td>15.7</td>
<td>17.2</td>
<td>19.0</td>
<td>23.2</td>
<td>26.8</td>
<td>29.2</td>
<td>30.8</td>
<td>31.7</td>
<td>33.3</td>
<td>6.9:1</td>
</tr>
<tr>
<td>Lena‡</td>
<td>15.6</td>
<td>16.6</td>
<td>17.7</td>
<td>21.3</td>
<td>23.9</td>
<td>25.9</td>
<td>27.4</td>
<td>28.6</td>
<td>35.5</td>
<td>5.6:1</td>
</tr>
<tr>
<td>Jennifer†</td>
<td>14.8</td>
<td>16.1</td>
<td>17.4</td>
<td>20.8</td>
<td>23.1</td>
<td>24.9</td>
<td>26.3</td>
<td>27.6</td>
<td>35.6</td>
<td>5.5:1</td>
</tr>
<tr>
<td>Pyramid†</td>
<td>15.8</td>
<td>17.2</td>
<td>18.6</td>
<td>21.0</td>
<td>23.0</td>
<td>24.8</td>
<td>26.4</td>
<td>27.7</td>
<td>31.5</td>
<td>5.4:1</td>
</tr>
</tbody>
</table>

Table 3: PSNR of Intermediate Images in Decoding Process

‡ \( \alpha_{\text{max}} = 1.5 \); † \( \alpha_{\text{max}} = 1.0 \)
BLOCK ARTIFACTS

Plates 6, 7, and 8 display a comparison between the original images and their fractal approximations. The enhanced difference images are an inverted representation of the absolute difference between the images multiplied by eight. We can see that block artifacts show up on Lynn's cheek bones and her nose. Blockiness also shows up in Lena's shoulder and the brim of her hat. This is due to the classification of those blocks as shade blocks. Some of this objectionable blockiness can be reduced by increasing the image block classifier's sensitivity to mild gradient, and thus encoding these blocks as midrange blocks. Unfortunately, this will also increase the number of bits per pixel and the encoding time. With the exception of shade blocks, very little blockiness is evident. Note that Jennifer, with few shade blocks, shows very few block artifacts and attains a high PSNR.

CONTRACTIVITIES REVISITED

In this algorithm, contraction is obtained in one of two ways: shade block encoding ($s = 0$) and midrange or edge block encoding where $\alpha < 1$. When the $\alpha_{\text{max}}$ is set greater than 1.0, individual transformations can become expansive. The reason this does not necessarily cause a partition to fail to properly converge is that most of the blocks within a partition are interdependent by virtue of the geometric operator $S$ (spatial contraction). In many cases, this allows the contractive blocks to rein in the expansive blocks. Unfortunately, to the author's knowledge, this cannot be guaranteed.

In [5], Jacobs et al. note “Results... indicate that $s_{\text{max}} = 1.2$ or 1.5 usually yields the best PSNR versus compression results.” They continue, “It is of interest that every one of the encodings with $s_{\text{max}} \geq$
1.0 (numerous images and several hundred separate encodings) converged to a fixed point." Plates 10 and 11 provide illustrations of images which fail to properly converge with a $\alpha_{\text{max}} = 1.5$. These images are not standard test targets and originated in the author's personal photo library. The Mayan pyramid was photographed on Kodak Gold 100 film; Jennifer was photographed on Kodak Gold 200 film; and both were scanned with a Kodak RFS2035 Film Scanner. The dye cloud formations (grain) are recorded as high frequency noise in the image. This noise is very difficult for the algorithm to reproduce and may be the precipitating factor for the failure.

Consider an image containing a high frequency noise component (single pixel events). When a domain block is spatially contracted, it is essentially low-pass filtered. This will very likely decrease the block's dynamic range. In order to match the dynamic range of a range block, it is likely that a very high contrast scale factor will be required. When this happens frequently in an image, one or more partitions may cease to be contractive.

For each of the four partitions (numbered right to left, top to bottom) of each image encoded with $\alpha_{\text{max}} = 1.5$, Table 4 lists the average contractivity of each class of blocks. This value reflects no information on the interdependency of groups of blocks, only the average contractivity within the partition. It is interesting that even with total average contractivities above 1.0, some image partitions still converge properly.

The average contractivity of the midrange blocks in the two partitions which fail to converge is quite high. We see in Table 5 that most of the blocks in these partitions are midrange blocks. Moreover,
we see how the low-pass filter effect of the spatial contraction affects the distribution of block classes.

For most image partitions, the distribution of domain blocks into block classes is roughly the same as the distribution of range blocks. For the two partitions here which fail to converge, the distribution is significantly altered in such a way as to force a great number of range blocks to be dependent on far fewer domain blocks. This is counter to the ideal situation of homogenous interdependency.

<table>
<thead>
<tr>
<th>Name</th>
<th>Part.</th>
<th>Shade</th>
<th>Midrange</th>
<th>Edge</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lynn</td>
<td>1</td>
<td>0.000</td>
<td>1.066</td>
<td>1.076</td>
<td>0.520</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.000</td>
<td>0.967</td>
<td>0.987</td>
<td>0.509</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.000</td>
<td>1.011</td>
<td>1.307</td>
<td>0.615</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.000</td>
<td>0.961</td>
<td>1.057</td>
<td>0.609</td>
</tr>
<tr>
<td>Lena</td>
<td>1</td>
<td>0.000</td>
<td>1.180</td>
<td>0.894</td>
<td>1.059</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.000</td>
<td>0.969</td>
<td>0.861</td>
<td>0.749</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.000</td>
<td>0.984</td>
<td>0.949</td>
<td>0.907</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.000</td>
<td>0.800</td>
<td>0.847</td>
<td>0.620</td>
</tr>
<tr>
<td>Jennifer</td>
<td>1*</td>
<td>0.000</td>
<td>1.355</td>
<td>0.828</td>
<td>1.232</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.000</td>
<td>1.200</td>
<td>0.822</td>
<td>1.176</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.000</td>
<td>1.049</td>
<td>0.986</td>
<td>1.021</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.000</td>
<td>1.117</td>
<td>1.063</td>
<td>1.091</td>
</tr>
<tr>
<td>Pyramid</td>
<td>1*</td>
<td>0.000</td>
<td>1.481</td>
<td>0.896</td>
<td>1.037</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.000</td>
<td>1.074</td>
<td>0.981</td>
<td>0.909</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.000</td>
<td>1.191</td>
<td>0.955</td>
<td>1.126</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.000</td>
<td>1.272</td>
<td>0.993</td>
<td>1.187</td>
</tr>
</tbody>
</table>

Table 4: Average Contractivity by Block Class and Partition. * indicates partitions which failed to properly converge. 
\( \alpha_{\text{max}} = 1.5 \) for all images
Table 5: Block Class Distribution by Partition for Pyramid.

PSNR AND SUBJECTIVE IMAGE QUALITY

Although this paper does not implement the full algorithm described by Jacquin, it produces images of equal or better image fidelity. The purpose of parent/child encoding in the full algorithm is to improve compression rate and encoding speed. Recall that coded parent blocks (8x8) are decomposed into independently coded child blocks (4x4), if the distortion of the child blocks within a parent block are not below a threshold. The threshold value provides an opportunity for a less than optimal encoding. In this paper, we encode each block (4x4) optimally.

In this section we see how the subjective image quality and fidelity of Fractal Block Encoding compares to JPEG compression. Each of the test images used in this paper was encoded with five different levels of JPEG compression. The levels of compression chosen provide PSNR performance bracketing that obtained with fractal block encoding. These five images, along with one fractal
encoded image were used in a comparison study. Participants were asked to subjectively rank the images in order of worst to best. For each image, the high and low scores were thrown out and the average was computed. Table 6 illustrates the results of that ranking.

Subjective ranking closely followed the PSNR for JPEG images, but the fractal block encoded image scored slightly higher for some images than would be expected. Aggressive JPEG compression requires a the coarse quantization or outright elimination of many terms of the Discrete Cosine Transform (DCT) at the heart of this compression scheme. The distortion in the recovered image resulting from this loss in terms appears often as “ringing” around edges. This periodic ringing, an artifact of the sinusoidal basis function of the transform, is easily detected by the eye. The noise introduced by fractal block encoding is less periodic and not as easily detected.

Fractal block encoding is not as flexible as JPEG in providing a range of compression and corresponding PSNR rates. In the study by Jacobs et al [5], the best PSNR obtained from fractal block encoding was 35.9 dB. The range of PSNR obtainable from JPEG on our test images extended to 58.9 dB while providing a compression ratio of roughly 1.4:1.

Elimination of shade block encoding, while not improving PSNR, may significantly improve the subjective image quality by diminishing blockiness in the fractal block encoded image.
### Table 6: Comparison of PSNR (dB), Compression Ratio, and Subjective Image Quality

* Fractal Block Encoded Image

**Compression Ratios**

As implemented here, this algorithm compresses images roughly 6:1. Improved compression rates may be obtained by implementing any of the following:

- **Multi-level encoding.** Implement parent/child encoding used by Jacquin or the multi-level quad-tree encoding used in [5] by Jacobs et al. Use of the larger blocks allows us to encode large areas more efficiently.

- **Quantize the values of α and Δg to fewer values.** This may result in image degradation, but will save bits for each midrange and edge block encoded.

- **Encode smaller partitions.** This will require fewer bits to specify the address of the domain block, saving bits on each midrange and edge block. The drawback is that the pool of potential matching domain blocks is made smaller. This may be acceptable since most of the matching domain blocks are found close to the range block.

- **Entropy encoding.** Currently no entropy encoding is used. Excellent candidates for this include block type bits (only 3 of 4 combinations are currently used), ±distance to domain block, α, Δg, etc.
OTHER RESULTS

In [1], two level parent/child block encoding is used with 2 bits for encoding $\alpha$ and 6 bits for encoding $\Delta g$. Using those parameters, Jacquin claims to encode Lena at a rate of 0.06 bpb (16.7:1) with a PSNR of 31.4 dB.

In [5], Jacobs et al. use an encoding algorithm slightly different from Jacquin. Their encoding scheme expands the parent/child relationship to a quad-tree which begins with 32x32 pixel blocks. These blocks are recursively decomposed into 16x16, 8x8, and 4x4 blocks if the encoded block in question does not meet a distortion threshold. In addition, they include domain blocks which lie at a 45° angle to the natural boundaries of the image, and use a image block classification scheme different from the one used here. They publish a complete comparison of the effects of varying parameters such as $\alpha_{\text{max}}$, the quantization of $\alpha$ and $\Delta g$, the size of the domain block pool. For values similar to those used here, they obtained an compression rate of roughly 17:1 and a PSNR of 33.3 dB.

Conclusion

The Fractal block encoding technique proposed by Jacquin is an interesting alternative to JPEG encoding for compression of images. Multilevel encoding provides compression rate and PSNR performance comparable to JPEG. With entropy encoding, performance may improve further. Unfortunately the time to fractally encode an image is about 300 - 400 times that required to JPEG encode the same image at about the same PSNR. The time to decode a fractal encoded image, while almost two orders of magnitude
less than the encoding time, is still several times greater than the time to decode a JPEG encoded image.
References


Appendix A: Source Code

Note: All code was written using standard ANSI C library routines. Executable versions were created and run on Apple Macintosh® computers using the Symantic Think™ 5.0 compiler, and Sun Microsystems® workstations using the GNU gcc compiler version 2.5.8.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>common.h</td>
<td>General definitions.</td>
</tr>
<tr>
<td>block.h</td>
<td>Definitions and prototypes for block classifier, block transforms, and block distortion measure.</td>
</tr>
<tr>
<td>block.c</td>
<td>Code for block classifier, block transforms, and block distortion measure.</td>
</tr>
<tr>
<td>dec.c</td>
<td>Main function for fractal decoder.</td>
</tr>
<tr>
<td>enc.c</td>
<td>Main function for fractal encoder.</td>
</tr>
<tr>
<td>frAnalyse.c</td>
<td>Main function for fractal coded image analyser. Give it the name of a fractal image file and a block location, it will look up the encoding and display it in human readable form.</td>
</tr>
<tr>
<td>frStat.c</td>
<td>Main function for fractal coded image statistics generator. Output from this program was fed into Microsoft Excel to produce figures 3 through 6.</td>
</tr>
<tr>
<td>frac.h</td>
<td>Definitions and prototypes for fractal image encoder and decoder functions.</td>
</tr>
<tr>
<td>frac.c</td>
<td>Code for fractal image encoder and decoder functions.</td>
</tr>
<tr>
<td>iclss.c</td>
<td>Main function for interactive image block classifier. Give it a TIFF file and specify block coordinates and block size, it will print the hex values of the block, a representation of the quantized gradient in each the X and Y directions, statistics on the block, and its block class.</td>
</tr>
<tr>
<td>Tiff.h</td>
<td>Defines specific to TIFF file formats.</td>
</tr>
<tr>
<td>image.h</td>
<td>Defines and prototypes for reading and writing TIFF and (RIT) IMG file formats.</td>
</tr>
<tr>
<td>File</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>image.c</td>
<td>Code for reading and writing TIFF and (RIT) IMG file formats.</td>
</tr>
<tr>
<td>imgdif.c</td>
<td>Main function for calculating the PSNR, RMS error, and the difference between two images of similar dimensions.</td>
</tr>
<tr>
<td>rwBits.h</td>
<td>Defines and prototypes for functions designed to simplify the reading and writing of bit streams to a file. Allows the programmer to disassociate the bit stream from its byte alignment.</td>
</tr>
<tr>
<td>rwBits.c</td>
<td>Code for functions designed to simplify the reading and writing of bit streams to a file. Allows the programmer to disassociate the bit stream from its byte alignment.</td>
</tr>
<tr>
<td>tstBits.c</td>
<td>Main function for testing rwBits components. A cross-platform development tool.</td>
</tr>
</tbody>
</table>
Plate 1: Comparison of Partial Encoding Techniques

Original Image
Shade Block Encoding Only

Shade and Midrange Encoding
Full Fractal Encoding

All images printed at 125 pixels per inch
Plate 2: Iterations of Fractal Image Decoder for Lynn
All images printed at 250 pixels per inch.
Plate 3: Iterations of Fractal Image Decoder for Lena

All images printed at 250 pixels per inch.
Plate 4: Iterations of Fractal Image Decoder for Jennifer
All images printed at 250 pixels per inch.
Plate 5: Transmography Pyramid into Lena
All images printed at 250 pixels per inch.
Plate 6: Fractal Reconstructed Image vs. Original (Lynn)

Original Image

Fractal Decoded Image

Enhanced Difference Image

All images printed at 125 pixels per inch
Plate 7: Fractal Reconstructed Image vs. Original (Lena)

Original Image

Fractal Decoded Image

Enhanced Difference Image

All images printed at 125 pixels per inch
Plate 8: Original and Fractal Decoded Image for Jennifer

Original Image

Fractal Decoded Image

Enhanced Difference Image

All images printed at 125 pixels per inch
Plate 9: Fractal Reconstructed Image vs. Original (Pyramid)

Original Image

Fractal Decoded Image

Enhanced Difference Image

All images printed at 125 pixels per inch
Plate 10: Divergent Fractal Image vs. Original (Pyramid)

Original Image
Fractal Decoded Image

Enhanced Difference Image

All images printed at 125 pixels per inch
Plate 11: Iterations of an Unstable Fractal Code
All images printed at 250 pixels per inch.

Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 5
Iteration: 6
Iteration: 7
Iteration: 8
Iteration: 16
Iteration: 32
Stopped at Iteration: 128