A Hybrid Fractal-Wavelet Transform Image Data Compression Algorithm

Ian Levy, Roland Wilson
Department of Computer Science,
University of Warwick,
Coventry

September 1, 1995

Abstract

This report describes two seemingly distinct areas of work, wavelet analysis and fractal image compression. A review of these two areas is presented, a new algorithm outlined, and some results presented. Finally, some speculations concerning the future direction of this research is included.

Contents

List of Figures

1 Introduction

The work in this paper can be divided into two main areas. The first is fractal image coding and the second is wavelet analysis. The concept that unifies these two otherwise disparate areas is *invariance*. Both fractal coding techniques and wavelet analysis exploit scale invariance. Fractal coding exploits similarities between an image and a spatially averaged copy, while wavelets demonstrate the scale invariance of edges in an image. We shall now cover the background of both of these areas of research.

2 Fractal Block Coding

Fractal block coding is based on the ground breaking work of Barnsley [1] [2] [3] and was developed to a usable state by Jacquin[11]. The basic concept underlying this technique is that for each image, there exists a block-wise transform upon the image that will leave the image unchanged. The roots of fractal block coding lie in the mathematical world of metric spaces and, in particular, the Contraction Mapping Theorem.

2.1 The Contraction Mapping Theorem

Definition 2.1 (Contraction Mapping) *Let (X, d) be a complete metric space. That is let X be a vector space over some field $\mathcal{F}$ and d be a metric upon X. Then, a transformation T, $T : X \to X$ is said to be a* contraction mapping *iff*

$$\exists s \in \Re, 0 \leq s < 1 \;\; s.t \;\; d(T(x), T(y)) \leq s \cdot d(x, y) \; \forall x, y \in X. \tag{1}$$

Here, s is known as the *contractivity factor* of the transformation T .

Theorem 2.1 (Contraction Mapping Theorem) *For a contractive transformation T on a complete metric space X, there exists a* unique *fixed point $x^\star \in X$ such that $x^\star = T(x^\star)$. The unique fixed point is then*

$$x^\star = \lim_{i \to \infty} T^i(x_0) \; , x_0 \in X$$

Fractal coding of an image, x, is based upon finding a transformation T such that $d(x, x^\star)$ is minimised. However, in practice, $x^\star$ is not known and we instead seek to minimise the measure $d(x, T(x))$. This sub-optimal choice is justified by the *Collage Theorem* (Theorem 2.2).

2.2 Iterated Function Systems and The Collage Theorem

The term *iterated function system* was first used by Barnsley and Demko[2]. We define an iterated function system (IFS) thus.

Definition 2.2 (Iterated Function System) An iterated function system consists of a complete metric space (X, d) together with a finite set of contraction mappings $\{w_n\}$, $w_n : X \to X$ with contractivity factors s_n, $n = 1, 2, \ldots, N$. The notation for this IFS is $\{X; w_n,\ n = 1, 2, \ldots, N\}$ and its contractivity factor is $s = \max\{s_n,\ n = 1, 2, \ldots N\}$.

Then, according to Barnsley's IFS Theorem[3], the transformation $W : \mathcal{H}(X) \to \mathcal{H}(X)$, defined by

$$W(B) = \bigcup_{n=1}^{N} w_n(B), \forall B \in \mathcal{H}(X)$$

is a contraction mapping with contractivity factor s. In particular, W has a unique fixed point, $A \in \mathcal{H}(X)$, given by $A = \lim_{n\to\infty} W^n(B)$. Here, $\mathcal{H}(X)$ is the Hausdorff space of X, that is the set of all compact, non-empty subsets of X. To make $\mathcal{H}(X)$ into metric space, we must then define the *Hausdorff metric* as $h(A, B) = \max\{d(A, B), d(B, A)\}$ where $d(A, B) = \max\{d(x, B)\ :\ x \in A\}$. We now have the machinery to state the Collage Theorem proper.

Theorem 2.2 (The Collage Theorem) *Let (X, d) be a complete metric space and let $B \in \mathcal{H}(X)$ and $\epsilon \geq 0$. Choose an IFS $\{X; w_0, \ldots w_N\}$ with contractivity factor $0 \leq s < 1$ such that*

$$h\left(B, \bigcup_{n=0}^{N} w_n(B)\right) \leq \epsilon,$$

where $h(d)$ is the Hausdorff metric. Then,

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

For our purposes, the Collage theorem states that if we wish to find an IFS with an attractor that is close to (in the Hausdorff sense) a given image, $\mathcal{I}$, we may instead find a set of contractive transformations on compact subsets of $\mathcal{I}$ such that the union of the set of transformations applied to $\mathcal{I}$ is Hausdorff-close to $\mathcal{I}$. This is how standard fractal block coders work. That actual algorithm shall be covered next.

2.3 Conventional Fractal Block Coding

Take an arbitrary image $\mathcal{I}$ and partition it into non-overlapping blocks, $\{D_j\}$ of size $d \times d$. These blocks will be known as *domain blocks* , following the notation[1] of Barnsley[3]. We then spatially average the image by 2 in both the horizontal and vertical directions, extracting all, possibly overlapping, blocks of size $d \times d$ producing

[1]Barnsley's notation is slightly confusing since it refers to the *inverse* transform.

Ordinal	*Isometry*
1	Identity
2	Flip along mid-X axis
3	Flip along mid-Y axis
4	Flip along major diagonal

Table 1: Base isometry set used in conventional fractal block coders

the pool of *range blocks*, $\{R_i\}$. The goal of fractal block coders is then to find the optimal parameter set $\{a, b, c, i\}$ for each block D_j in the approximation

$$\hat{D}_j = a \cdot \iota_c(R_i) + b$$

such that the error $d(D_j, \hat{D}_j)$ is minimised. It is usual for d to be the metric derived from the L_2 norm, $\|D_j - \hat{D}_j\|^2$. Here, ι_c is an isometry generated from the set shown in Table 1. Since these four basic isometries generate the dihedral group , we may combine these to form any of the eight possible symmetries of the block.

The fractal block code for the image then consists of the parameter set $\{a, b, c, i\}$ for each domain block D_j in the domain pool $\{D_j\}$. The parameter sets are entropy coded to provide compression. We note now that the parameter set $\{a, b, c, i\}$ is restricted since the final transform T_j must be a contraction mapping under the metric d. We note from Jacquin[11] that the isometries all have L_2-contractivity of 1 and the contrast scale by a has L_2-contractivity of a^2. Hence, to ensure that the transform for each block is a contraction mapping, we must restrict the value of a such that $0 \leq a < 1$. Then, by the Collage Theorem (2.2), the union of these block transforms is itself a contraction mapping.

2.4 Reconstructing an Image from a fractal block code

We note that since $T = \cup_i T_i$ is a contraction mapping, by the Contraction Mapping Theorem, it has a unique fixed point. Since each component transform was chosen to minimise the error between the reconstruction and a block in the original image, the fixed point of the union of these transforms is 'close' to the original image. The decoding may be started from any initial image (since $\forall x \in X, T^n(x) \rightarrow x^*$ as $n \rightarrow \infty$). However, it is usual to start with a uniform grey image. Reconstruction proceeds by applying each component map to the initial image to generate the next image in the reconstruction sequence. We then spatially average this reconstructed image and apply the component maps again. This iterative procedure is repeated until there is little difference between consecutive images in the reconstruction sequence, or until some error condition has been satisfied.

2.5 Advances in Fractal coding

Fractal coding is an asymmetrical process; that is, coding takes much longer than decoding. Much research has revolved around speeding up the coding in some way. Jacquin[11] himself suggests classifying the range blocks into three distinct classes: shade blocks, midrange blocks and edge blocks. The coder then only checks range blocks in the same class as the current domain block when searching for the optimal transform. Details of the classification algorithm used by Jacquin can be found in [20].

Jacobs, Fisher and Boss [10] classify blocks into 72 different classes and also apply a quadtree partitioning scheme. The quadtree scheme works by using larger blocks (32×32 in their paper) and splitting the block into four smaller blocks should an error condition not be satisfied. This quadtree decomposition is repeated until the error condition is satisfied or a minimum block size is reached. This has obvious advantages if parts of the image contain large regions of relatively constant greyscale.

The same authors [9] proved that not all the w_i in the IFS $\{X; w_0, \ldots, w_n\}$ need to be contractive. In this report, they define a map $W : F \to F$ as *eventually contractive* if, for some $n \in \mathcal{Z}^+$,W^n is contractive. They then prove that the fractal decoder will converge if $T = \cup_i T_i$ is only eventually contractive. Here, we note that the iterated transform T^m is the composition of transform unions of the form

$$w_{i_1} \circ w_{i_2} \circ \ldots \circ w_{i_m}.$$

Since the contractivities of each union w_{i_j} multiply to give the overall contractivity of the iterated transform, the composition may be contractive if it contains sufficiently contractive w_{i_j}. Intuitively, it is simple to see that if the union consists of slightly expansive transforms and highly contractive ones, then the union will eventually be contractive. The provision for eventually contractive maps allows the coder to achieve better results. Since there is now no longer a bound on the contractivity of the component transform , the dynamic range of range blocks may now be *increased* to be similar to that of the domain block.

Most speed ups have, in some way, reduced the size of the pool that the coder must search for each domain block. Reducing the size of the search pool, however, can have an adverse effect on reconstructed image quality since the range block pool is not as rich. Saupe [21] uses the theory of multi-dimensional keys to perform an approximate nearest-neighbour search. Since this search can be completed in $O(logN)$ time, the range pool need not be depleted to achieve a speed up. Saupe's basic idea is that of a $(d-1)-$dimensional projection on $\Re^d$ where $d \geq 2$. He defines a subspace, $X \subseteq \Re^d$ as $X = \Re^d \backslash \{re : r \in \Re\}$ where $e = \frac{1}{\sqrt{d}}(1, \ldots, 1) \in \Re^d$. Defining a normalised projection operator $\phi : X \to X$ and a function $D : X \times X \to [0, \sqrt{2}]$ by

$$\phi(x) = \frac{x - < x, e > e}{\|x - < x, e > e\|}$$

and

$$D(x,z) = \min(d(\phi(x),\phi(z)), d(-\phi(x),\phi(z)))$$

gives an expression for the least squares error

$$E(x,z) = < z, \phi(z) >^2 g(D(x,z))$$

where $g(D) = D^2(1 - D^2/4)$. This theorem states that the least squares error, $E(x,z)$ is proportional to $g(D)$ which is a simple function of the Euclidean distance between $\phi(x)$ and $\phi(z)$ (or $-\phi(x)$ and $\phi(z)$ since $\phi(x)$ is unique up to sign). We also note that g is monotonically increasing on the interval $[0, \sqrt{2}]$. Saupe then states that the minimisation of the least squares errors $E(x_i, z)$ $i = 1, \ldots, N$ is equivalent to the minimisation of the $D(x_i, z)$. Thus, we may replace the least squares error minimisation by the nearest neighbour search for $\phi(z) \in \Re^d$ in the set of $2N$ vectors $\pm\phi(x_i) \in \Re^d$. Since we are now in a Euclidean space, we may apply any of the nearest neighbour search algorithms that run in expected logarithmic time, for example kd-trees [7]. Saupe does, however, note that an 8×8 block results in 64 dimensions for the multi-dimensional keys. He therefore suggests downsampling the blocks to, say, 4×4 which reduces storage requirements to just 16 dimensions.

Other speed up methods have revolved around performing little or no searching for the range blocks. Monro et al [17] [16] [14] [15] , have developed and patented a technique known as the *Bath Fractal Transform* [2] (BFT). The BFT works by limiting the search of the range blocks and also using higher order functions on the blocks. Mathematically, if $W = \{X; w_k, k = 1, \ldots, N\}$ is an IFS with attractor A, they define a *fractal function* f on A as $f(w_k(x,y)) = \nu_k(x, y, f(x,y))$ where the maps ν_k have parameters $\alpha_i^{(k)}, k = 1, \ldots, N, i = 1, \ldots M$. Then, M is the *order* of the IFS and N is the number of free parameters of the BFT. The function f is found by minimising $d(g, \hat{g})$ for some suitable metric d over block k where

$$\hat{g}(x) = \nu_k(w_k^{-1}(x), g(w_k^{-1}(x))).$$

Solving

$$\frac{\partial d(g,\hat{g})}{\partial \alpha_i^{(k)}} = 0 \ \ \forall i, k$$

gives us the BFT. They define various searching options for the BFT, defined before downsampling has occurred; that is, they assume that domain blocks are $d \times d$ while range blocks are $2d \times 2d$. A level zero search chooses the range block in the same position as the domain block. A level one search would include the four range blocks that intersect the domain block. A level two search would also include all range blocks that intersect those in level one and so on. The complexity options they use, however, make the BFT unique. As it stands, such a limited search would severely degrade

[2] They are at Bath University, England.

reconstructed image quality. Allowing higher order terms in the BFT reduces the error while keeping encoding times low. At its most basic level (level zero), the BFT degrades to Jacquin's method[11]. That is, the maps ν_k have the form

$$\nu_k(x, y, f) = s_k \cdot f + t_k.$$

A level three complexity gives the maps the form

$$\nu_k(x, y, f) = a_3^{(k)} x^3 + a_2^{(k)} x^2 + a_1^{(k)} x + b_3^{(k)} y^3 + b_2^{(k)} y^2 + b_1^{(k)} y + s_k f + t_k.$$

Note that there are no cross product terms (for example xy or $x^2 y$) so that calculation is kept relatively simple. They have recently [18] developed a proprietary reconstruction algorithm, the *Accurate Fractal Rendering Algorithm* (AFRA) ,which remains unpublished at this time, specifically designed for use with the BFT.

Barthel and Voyé [4] have modified the luminance transform to act in the frequency domain. They propose the following high-order luminance transform.

$$\lambda(g) = IDCT\left(\bigcup_{u=0}^{N-1}\bigcup_{v=0}^{N-1} a(u,v) \cdot G(u,v) + b(u,v)\right)\ ,\ G(u,v) = DCT(g)$$

where DCT denotes the Discrete Cosine Transform and $IDCT$ denotes the inverse discrete cosine transform, N is the size of the blocks and g is the range block itself. If we denote the DCT of the domain block f by $F(u, v)$, we can approximate the spectrum of the domain block $F(u, v)$ by scaling of the spectrum of $G(u, v)$. They conjecture that most blocks can be sufficiently well approximated by a low order transform, hence negating the risk of a bit-explosion due to the excessive number of parameters to be coded. Further to this, they propose modifying the luminance transform so that it operates on a frequency domain partition. The luminance transform would then be expressed as

$$\lambda_k(g) = IDCT\left(\bigcup_{u=0}^{N-1}\bigcup_{v=0}^{N-1}\left\{\begin{array}{ll} a_0 \cdot G(u,v) + b & \text{if } u=0, v=0 \\ a(u,v) \cdot G(u,v) & \text{otherwise} \end{array}\right.\right)$$

where $a(u, v) = a_i$ if $(u, v) \in R_i$ $i = 1, \ldots, K$. This permits their modified frequency domain luminance transformation to be used in the block splitting procedure they use later. This is similar to quadtree partitioning, but if the top level block does not satisfy the error condition, they only recode the sub-blocks which do not satisfy the error condition. In this way, they can achieve similar reconstruction results to quadtree partitioning but with fewer transform coefficients.

Gharavi-Alkhansari and Huang [8] use a combination of fractal coding and basis block projection. For each domain block, they generate three pools of range blocks thus

i *Higher Scale Adaptive Basis Blocks*
This pool is the standard range block pool from a normal fractal coder. That is, spatially averaged copies of the domain blocks, augmented by rotated and reflected versions.

ii *Same Scale Adaptive Basis Blocks*
This pool is generated by selecting regions of the image which are the same size as the domain blocks. These blocks, however, must be selected causally, that is they may only be selected from parts on the image which have already been encoded. This pool may also be augmented by rotations and reflections.

iii *Fixed Basis Blocks*
This is a fixed pool of basis blocks that are known *a priori* to both the encoder and decoder. They need not be orthogonal or even complete.

The purpose of the fixed basis blocks is to allow the encoder to accurately encode a domain block that is totally dissimilar from any range block in the image. However, the lack of the orthogonality condition on the basis blocks appears to make the optimal solution for the coefficients of the basis blocks a difficult task. The authors offer two sub-optimal methods of calculating basis block coefficients. The first is to select the basis block most strongly correlated with the domain block. Then, orthogonalise the domain block with respect to the basis block. Repeat until an error condition is satisfied or all basis blocks have been used. The second, more general method, is to do the same as the first method, but also orthogonalise all other basis blocks with respect to the most correlated. They also note that standard fractal block coding, block transform coding and vector quantisation are all special cases of their proposed algorithm.

3 The Wavelet Transform

The Fourier Transform method is a well known tool for signal analysis. The Fourier Transform expresses any square integrable (in the Lebesgue sense) 2π periodic function on $L^2([0,2\pi])$ as the projection of it onto the orthonormal basis

$$\omega_n(x) = e^{\imath nx}, \quad n = \ldots, -1, 0, 1, \ldots.$$

However, noting that if $\omega(x) = e^{\imath x}$ then $\omega_n = \omega(nx)$. Hence, the orthonormal basis $\{\omega_n\}$ is actually the set of integer dilates of the single function ω. We note that

$$\omega(x) = e^{\imath x} = \cos x + \imath \sin x.$$

The remarkable fact about the Fourier Transform is that this is the *only* function required to generate *all* 2π periodic square integrable functions.

We now turn our attention to the more useful space $L^2(\Re)$. Functions in $L^2(\Re)$ satisfy the following condition

$$\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty.$$

Since every function in $L^2(\Re)$ must decay to zero at $\pm\infty$, the Fourier basis functions ω_n, which are sinusoids, are not in $L^2(\Re)$. A synonym for sinusoids is *waves* and if we require waves that generate $L^2(\Re)$, they must decay very quickly to zero (for all practical purposes). Hence, we need small waves or *wavelets* to generate $L^2(\Re)$. As in the Fourier Transform case, we would like the wavelets to be generated by one function. However, if the wavelets must decay to zero, we must translate the function dilates along $\Re$ by integral shifts. If we choose a *mother wavelet* $\psi \in L^2(\Re)$ of unit length, then all the translated dilates

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k)\ ,\ j, k \in \mathcal{Z}$$

also have unit length.

Definition 3.1 (Wavelet) *A function $\psi \in L^2(\Re)$ is called a* wavelet *if the family $\{\psi_{j,k}\}$ defined by*

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k)\ ,\ j, k \in \mathcal{Z}$$

is an orthonormal basis of $L^2(\Re)$. That is

$$< \psi_{j,k}, \psi_{l,m} >= \delta_{j,l}\delta_{k,m}\ ,\ j, k, l, m \in \mathcal{Z}$$

where

$$\delta_{j,k} = \begin{cases} 1 & \textit{if } j = k \\ 0 & \textit{otherwise} \end{cases}$$

Then, every $f \in L^2(\Re)$ can be written as

$$f(x) = \sum_{j,k=-\infty}^{\infty} c_{j,k}\psi_{j,k}(x). \tag{2}$$

The series representation of f in (2) is called the *wavelet series* of f and is analogous to the Fourier series of a function. Similarly, we may define the *integral wavelet transform* [5], which is analogous to the continuous Fourier Transform as

$$(W_\psi f)(b, a) = |a|^{-1/2} \int_{-\infty}^{\infty} f(x)\overline{\psi\left(\frac{x-b}{a}\right)} dx, \quad f \in L^2(\Re).$$

It is worth noting at this point a subtle, yet important difference between the Wavelet Transform and the Fourier Transform. Fourier basis functions are only localised in frequency, not in time. Small coefficient errors in the Fourier transform will produce changes everywhere in the time domain. Wavelets, however, are localised both in frequency(scale) by dilations *and* in time by translations.

3.1 The Discrete Wavelet Transform

The continuous wavelet transform is a useful theoretical tool, but in image processing, we deal with sampled images, i.e discrete values. Here, we would require a simple and fast method of calculating the wavelet coefficients of a signal. Wilson [24] defines the 1-D discrete wavelet transform in terms of a pair of *quadrature mirror filters* (QMF pairs). If $\{g_n\}$ is a sequence that is generated by sampling the mother wavelet ψ, then g will be a high pass filter. The mirror filter of this is $\{h_n\}$, given by the relation $g_m = (-1)^m h_{1-m}$. We then define the discrete wavelet transform in terms of successive filtering and dyadic downsampling.

$$\begin{aligned} c_n^i &= \textstyle\sum_k h_k c^{i-1}(2n-k) \\ d_n^i &= \textstyle\sum_k g_k c^{i-1}(2n-k) \end{aligned}$$

where $c^0(n) = f(n)$ is the original signal and the wavelet transform coefficients consist of $\{d_n^i\}$. For reconstruction, it is usual to zero pad the data with $2^i - 1$ zeros at level i. If we define a family of filters $\{h_n^i\}$ given by

$$h_k^i = \begin{cases} h_n & \text{if } k = 2^i n \\ 0 & \text{otherwise} \end{cases}$$

and let $f^i(n)$ be the zero padded data at level i, then the reconstruction from level i to level $i-1$ is given by

$$f^{i-1}(n) = \sum_k h_{-k}^{i-1} f^i(n-k).$$

The path from the 1-dimensional transform to the 2 dimensional version is simple. We note that, with the correct filters, the above definition of the discrete wavelet transform becomes separable in 2 dimensions. That is, we may perform the 1 dimensional wavelet transform on the rows, followed by the columns. Figure 1 and Figure 2 illustrate the 2 dimensional discrete wavelet transform.

In her landmark paper, Daubechies[6] develops a set of filters for orthonormal, compactly supported wavelets of differing sizes. These particular wavelet bases are ideal for block-wise image processing techniques since the orthonormality condition assures easy calculation while the compact support reduces the boundary artifacts at block edges. Figure 3 shows the Daubechies 8 point filter and Figure 4 shows the Lena image decomposed to three levels using this filter.

It is worth noting here that since the wavelet transform is a bounded linear operator, it preserves the norm on its underlying space. Hence, for our purposes of image coding, we may assume that the L^2 norm will be preserved across the wavelet transform. That is, $\|D_1 - D_2\|_2 = \|W(D_1) - W(D_2)\|_2$ where W is the wavelet transform.

Figure 1: Forward Discrete Wavelet Transform in 2 dimensions.

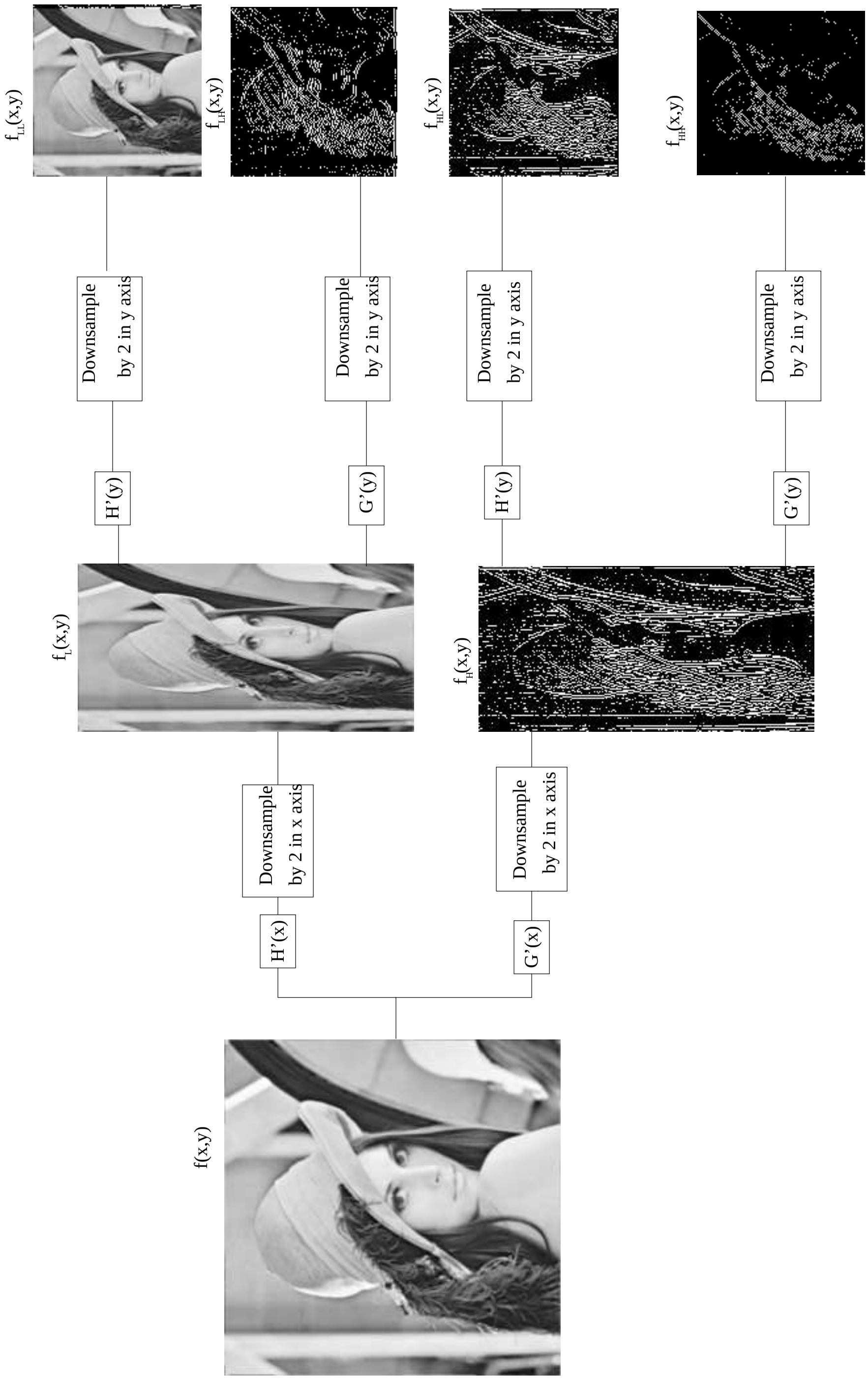


Figure 2: Inverse Discrete Wavelet Transform in 2 dimensions.

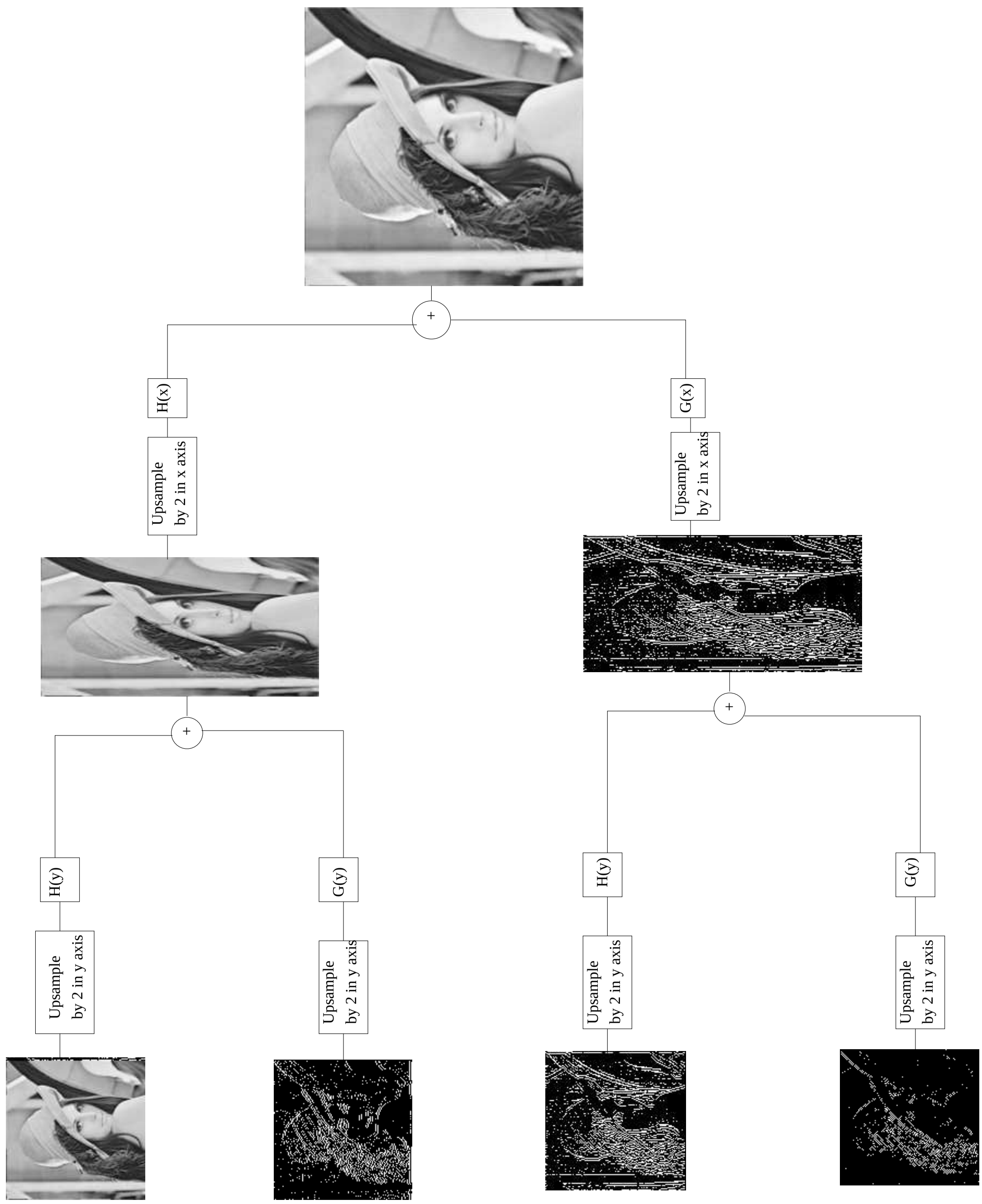


Figure 3: Daubechies 8 point wavelet filter

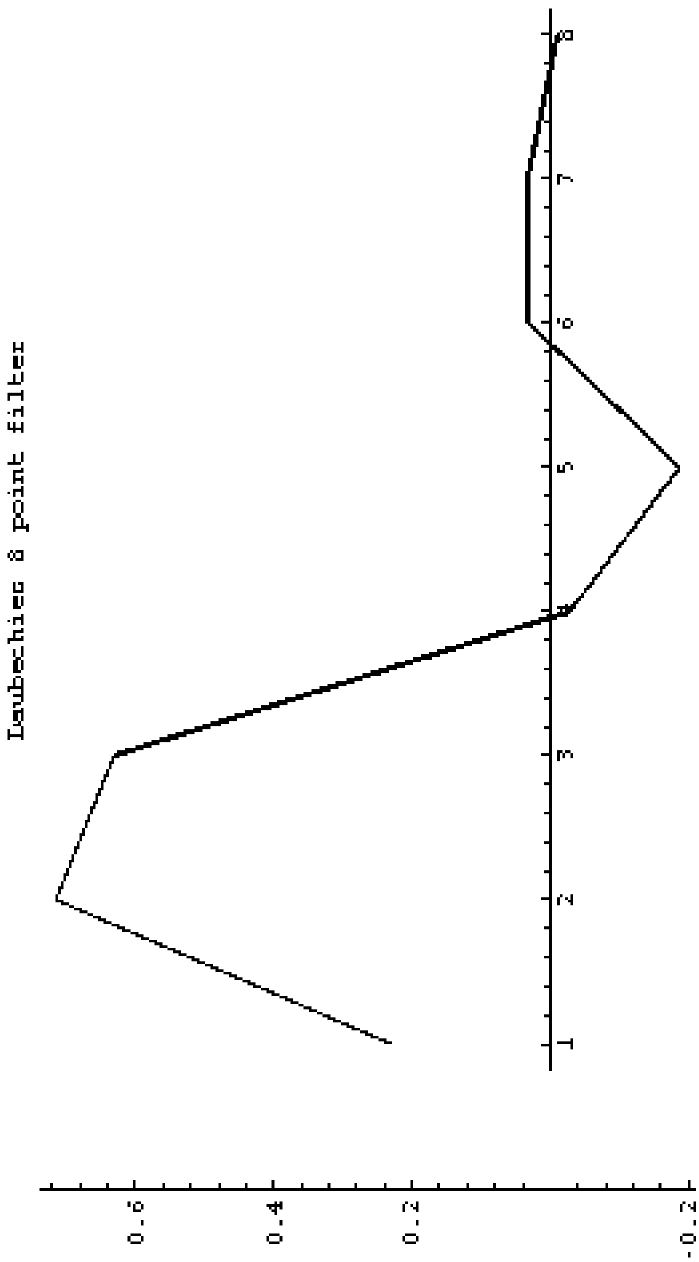
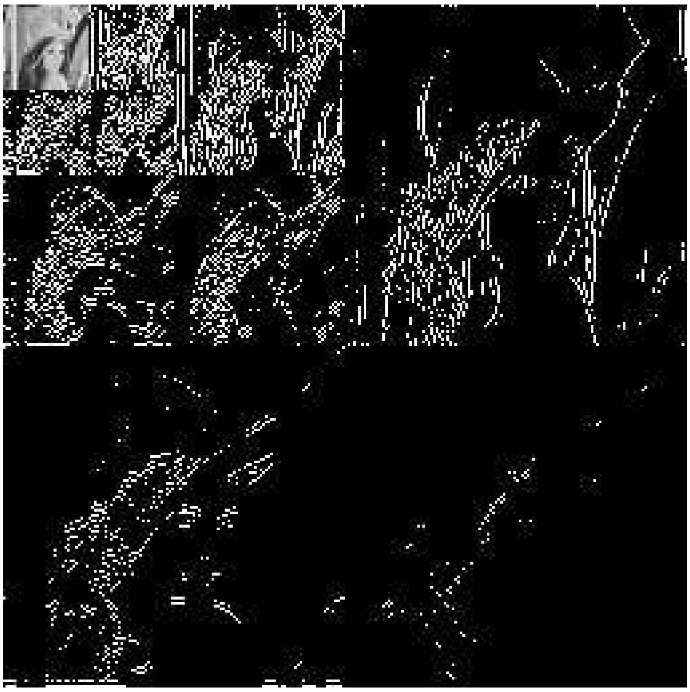


Figure 4: Three level Wavelet Decomposition of the Lena Image



3.2 Relation Between Iterated Function Systems and The Wavelet Transform

As discussed previously, iterated function systems rely of self similarity of compact subsets of $\Re^2$. In our case, these subsets support greyscale images. The wavelet transform is a tool for demonstrating the scale invariance of edges. Referring to Figure 4 , it is obvious that the sub-bands in successive wavelet levels are similar. It is, therefore, natural to move the iterated function system model to the wavelet domain. This exploits the scale invariance of edges to provide a form of approximate self-similarity required by the iterated function system model. Another advantage of the wavelet domain is that the mean of the coefficients is *scaled* across levels, not shifted. The coefficients are actually scaled by $1/\sqrt{2}$ by each of the 1-dimensional component transforms. Hence, the coefficients are scaled by $1/2$ for every level in the wavelet decomposition. Thus, no mean shift parameter is required for the iterated function system maps. Furthermore, there is an implicit scale factor of 2 in the block scale parameter of each map. We may divide this implicit value out and re-multiply at the decoder. This will compact the distribution of values and increase the scope for entropy coding.

Since the wavelet transform we are using is orthogonal, successive wavelet transforms are actually projections into orthogonal subspaces (see [6] and [13] for a description of the multiresolution analysis theory that embodies the orthogonal subspace property), we need not iterate the iterated function system maps at all. A single application is sufficient and this greatly reduces decoding time.

4 The New Algorithm

In this section we shall present a novel algorithm which performs fractal coding in the wavelet domain. This algorithm attempts to approximate a level of wavelet coefficients from the level above. It combines many areas of signal processing including wavelet transforms, fractal block coding, orientation estimates, standard block coding by basis blocks, quantisation and arithmetic coding. The specifics of each of these areas that are relevant to this algorithm will be covered in this section.

4.1 Algorithm Basics

To begin, we decide on the level of wavelet coefficients we wish to approximate. This will be known as the *domain level*. We also decide on the level from which we shall approximate which is usually the level above. This will be known as the *range level*. The image to be coded is decomposed by the wavelet transform to a given level, say l. Levels from l to the level below the range level are quantised using a linear quantiser for the high pass coefficients and a separate quantiser for the low pass coefficients at the highest level. An approximation to the range level is then generated via the inverse wavelet transform. We use the quantised approximation as the range level of coefficients to reduce errors at the decoder. The coder then treats each of the sub-bands of the wavelet decomposition separately. Each of the HL,LH and HH bands in the domain level are partitioned into non-overlapping domain blocks. Let $\{D_i^{HL}\}$, $\{D_i^{LH}\}$ and $\{D_i^{HH}\}$ be the domain blocks from each sub-band.

For each domain block set $\{D_i^{HL}, D_i^{LH}, D_i^{HH}\}$ we must find a corresponding range block set $\{R_i^{HL}, R_i^{LH}, R_i^{HH}\}$ which minimises the error $\|D_i^{HL} - \hat{D}_i^{HL}\|^2 + \|D_i^{LH} - \hat{D}_i^{LH}\|^2 + \|D_i^{HH} - \hat{D}_i^{HH}\|^2$ where $\hat{D}_i^X = \iota_i(\alpha_i \cdot \hat{R}_i^X) + b_i$. Here $\hat{R}_i$ is R_i normalised, $\alpha_i = \frac{<\hat{R}_i, D_i>}{<\hat{R}_i.\hat{R}_i>}$ and b_i is chosen to make the means of R_i and D_i equal. We note that since there will be no iteration of the maps in the decoding process, the value of α_i is unbounded, and, as stated in Section 3.2, there is no shift parameter necessary (so $b_i = 0 \forall i$). Given a domain block size d and and search radius r, we define the *search region* as the subset of the image $\mathcal{I}$ given by $([x - rd, x + rd] \times [y - rd, y + rd]) \cap \mathcal{I}$ where $\mathcal{I}$ is each of the three sub-bands. We begin a spiral search from the centre of the search region as in Figure 6. By using a spiral search, we may reduce the addressing problem to a single value. Figure 7 is a simplified flowchart of the coder. This method constrains the maps so that each range block is in the same relative position, but in different sub-bands. The intuitive reasoning behind this is that if two blocks are similar, then their horizontal, vertical and diagonal components must also be similar.

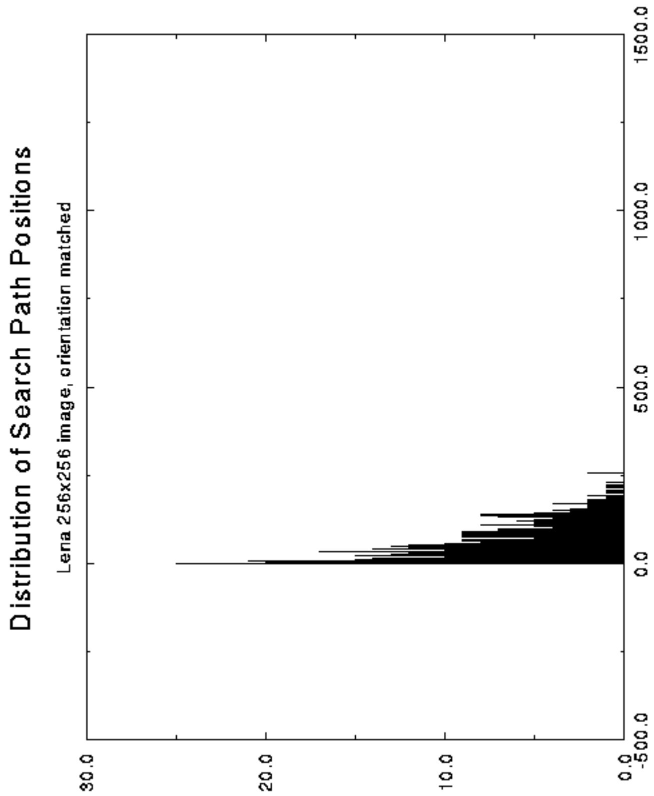


Figure 5: Distribution of path positions when block orientations are matched

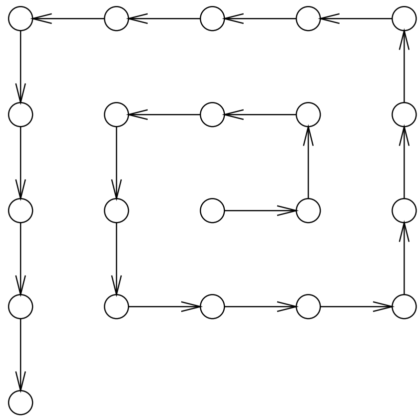


Figure 6: The spiral search path

4.2 The Reconstruction process

Reconstruction proceeds much the same as for standard fractal coding. Note that the range level of wavelet coefficients is transmitted to the decoder in some way. The major difference is that our method does not require iteration of the maps. The maps are applied once. An inverse wavelet transform from the domain level reconstructs the original image.

4.3 Coefficient Encoding

The parameter sets that describe the block-wise maps are linear quantised and encoded using an order zero arithmetic coder. We set a *rate value* which determines how many bins the quantiser uses and how many symbols the arithmetic coder can encode. Obviously, using less symbols increases compression but reduces image quality. There is a trade off to be found between image quality and compressed size. Figure 8 shows how the linear quantiser we use works. It is the same quantiser used by the JPEG system [19].

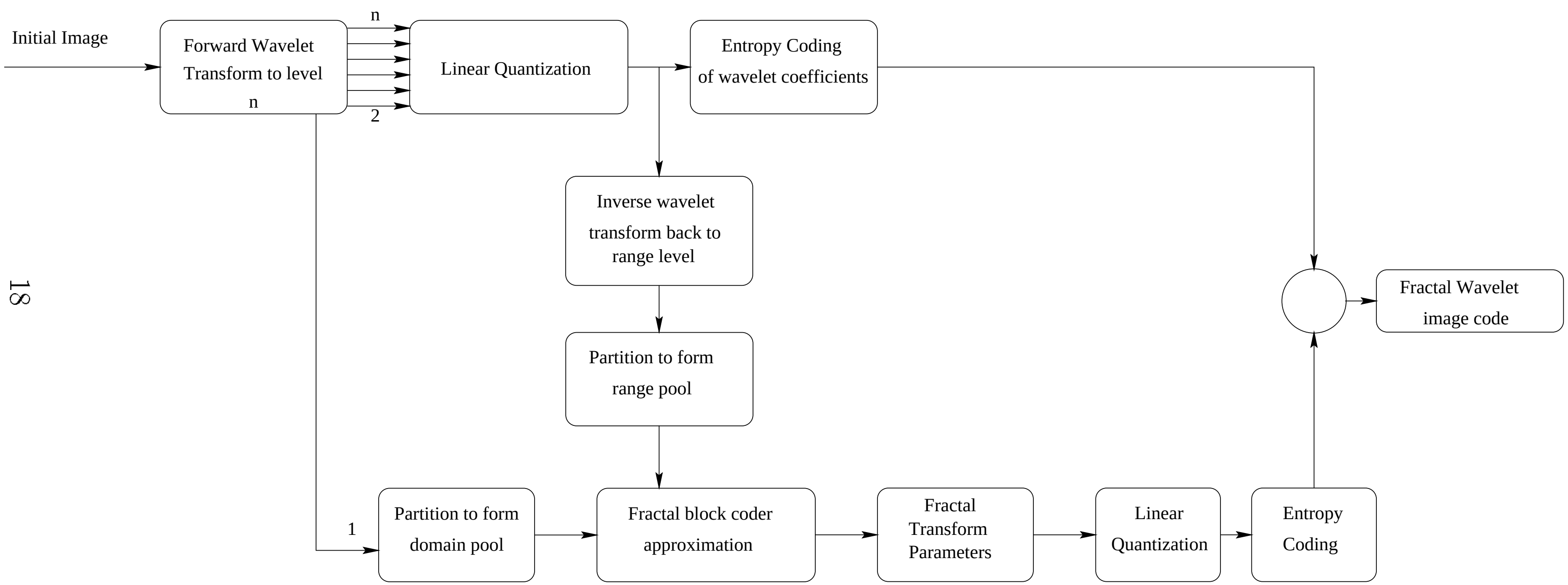


Figure 7: Block Diagram of the Basic Coder

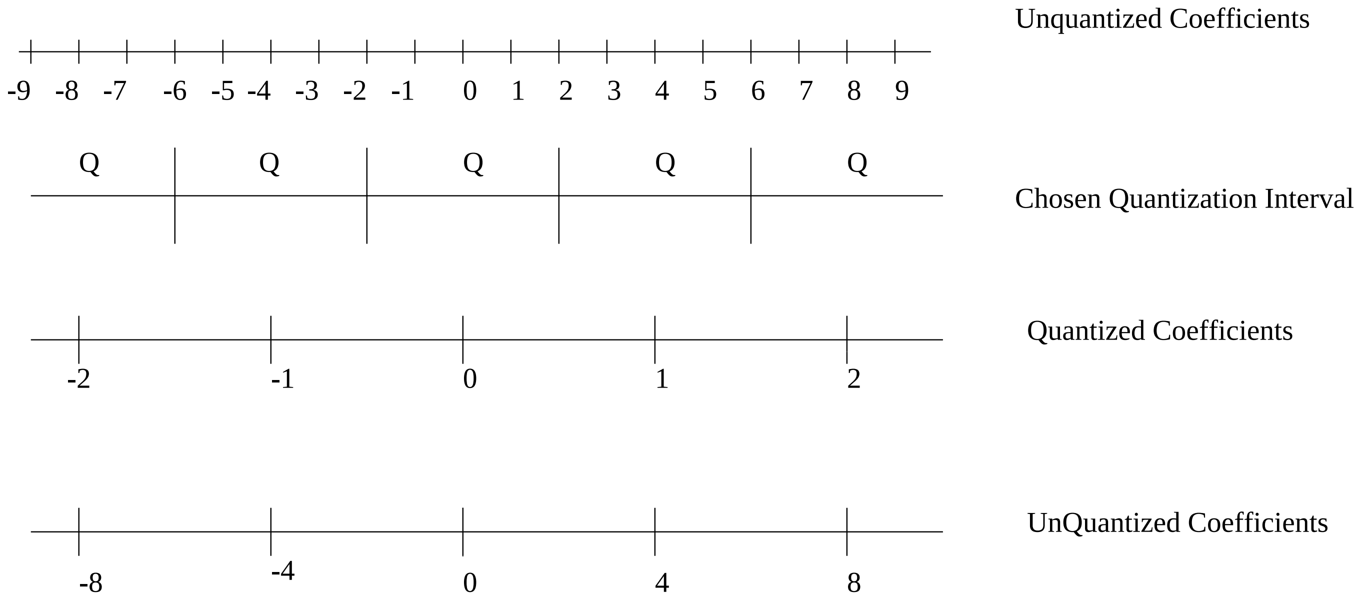


Figure 8: Quantising and Unquantising coefficients

Figure 9: Initial Code Interval for Arithmetic Coder

4.4 Arithmetic Entropy Coding

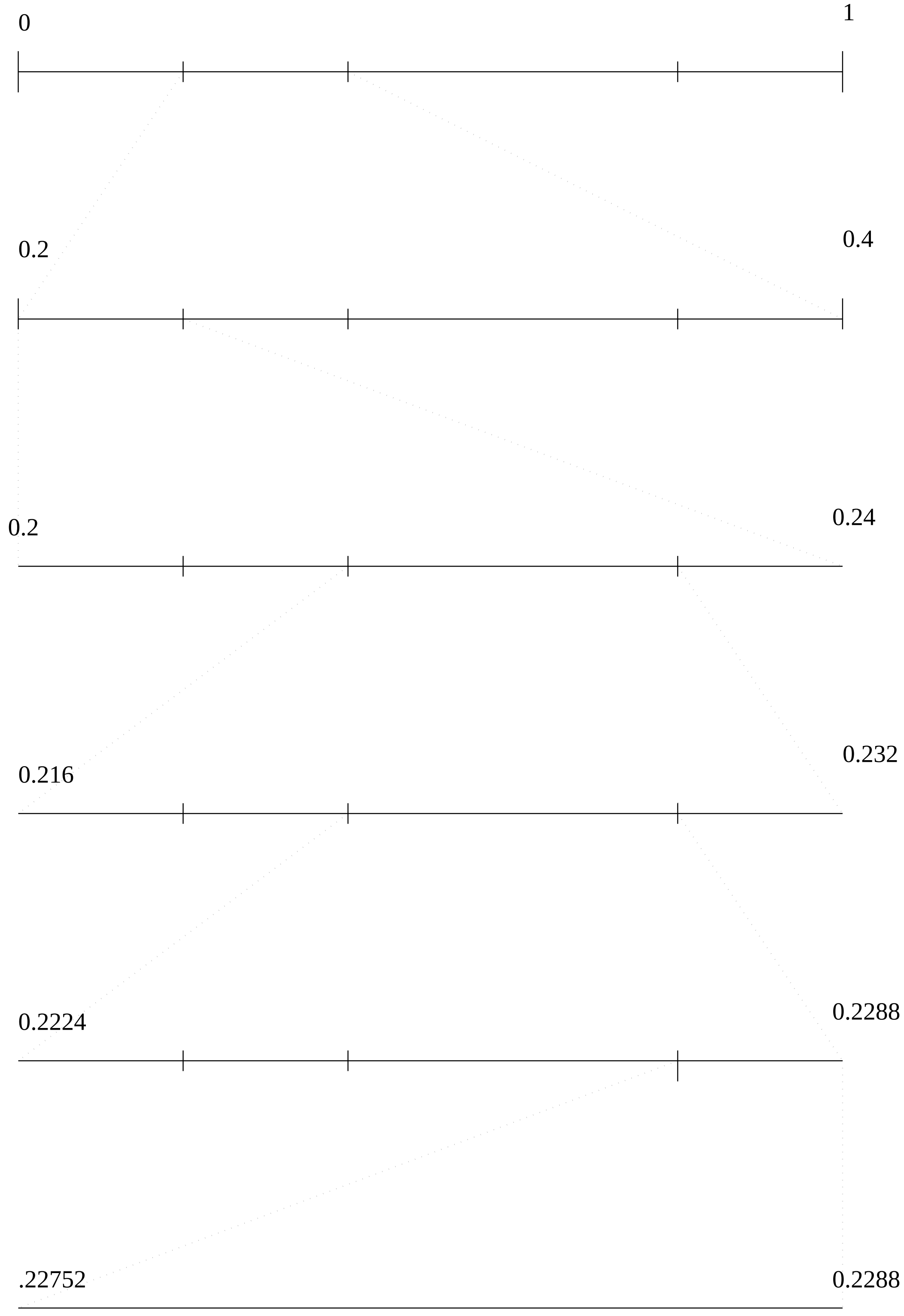
The goal of entropy coding is to reduce the number of bits required to encode a symbol, hence providing compression. Arithmetic coding, pioneered by Langdon [12] and popularised by Witten, Neal and Cleary [25], is based on the statistical properties of the signal. For example, if we have a symbol with probability of 123/456 of occurring, the goal of the Arithmetic coder would be to code the symbol in $-\ln(123/456)$ bits. There are many types of arithmetic coder, both adaptive and non-adaptive. The simplest adaptive case is the order zero model. This model generates the cumulative probabilities of symbols on the input stream. Other models include higher order models (i.e. those that predict probabilities over more than 1 symbol) and other, more complicated models. Williams [23] provides a good overview of the subject.

4.5 Operation of the Arithmetic Coder

The basic tenet of the arithmetic coder is that any sequence of symbols may be encoded in an arbitrary precision number. For the purpose of this example, let us assume we have the symbol space $\{e, h, l, o\}$ with probabilities $\{0.2, 0.2, 0.4, 0.2\}$. We shall begin with the interval $[0, 1]$, although any closed interval will suffice. We shall code the message *hello* using the arithmetic coder with a fixed probability as stated. Figure 9 shows how the coder would split the initial interval based on these probabilities.

Now, the coder receives the first symbol of the message. The symbol is 'h', which is assigned the code range $[0.2, 0.4]$. The code interval now becomes $[0.2, 0.4]$. Next, the coder receives the symbol 'e'. This is assigned the range $[0, 0.2]$ in the initial interval, so is assigned the range $[0.2, 0.24]$. Figure 10 demonstrates how the coder progresses. The final code interval is $[0.22752, 0.2288]$. Any number in this interval will successfully represent the message. It is, however, usual to use the midpoint. Arithmetic coding is plagued with caveats, such as range resolution and underflow. Williams [23] presents the problems and most of the common solutions.

Figure 10: Progression of the Arithmetic Coder



4.6 Extending the Algorithm

As it stands, the coder is still slow. To speed the coder up, we create an orientation map for the range blocks, using the double angle formula. Assume that we have the wavelet decomposition of the image at the range level, consisting of the bands f_{LL},f_{LH},f_{HL} and f_{HH}. Then the orientation of the pixel at position (x, y) in the LL band of the level below is approximated by

$$O(x,y) = \arctan\left(\frac{2 \times \left(f_{HL}(x,y) + \frac{f_{HH}(x,y)}{2}\right) \times \left(f_{LH}(x,y) + \frac{f_{HH}(x,y)}{2}\right)}{\left(f_{HL}(x,y) + \frac{f_{HH}(x,y)}{2}\right)^2 - \left(f_{LH}(x,y) + \frac{f_{HH}(x,y)}{2}\right)^2}\right). \quad (3)$$

These orientation estimates are then shifted to ensure they lie in the interval $[0, \pi/2]$ and quantised using a linear quantiser. This is possible since we shall apply rotational isometries to the blocks. A block's orientation is then simply defined as the average of the orientation vectors of the pixels in it. Furthermore, by not counting the blocks whose orientations defined by equation 3 do not match the quantised orientation of the domain block, we may compact the distribution of values, thereby increasing the scope for entropy coding. Figure 11 shows a block diagram of the modified coder.

Figure 5 and Figure 13 show the effect of not counting blocks whose orientations do not match.

As it stands, the coder attempts to extrapolate a block in the range level to a block in the domain level in each sub-band. However, if there is not a good match between wavelet levels, the overall reconstruction error will increase dramatically for each bad block. To overcome this, we perform block projection after the fractal coding to eliminate errors, in the same vein as Huang and Gharavi-Alkhansari [8]. Our method, however, is more simple. The basis we use is fixed and calculated over an assortment of images in the following manner. For each test image the difference between the domain block and its fractally generated approximation is stored for each band. When we have all error blocks, we symmetrise them using all the symmetries that the fractal coder may use. We now treat each $d \times d$ block as a d^2 dimensional vector. In our test cases, using a 4×4 block size results in a 16 dimensional vector. We then calculate the covariance matrix of all the symmetrised errors. This results is a $d^2 \times d^2$ dimensional matrix. Taking the eigenvectors of this covariance matrix and *folding* the vectors back to $d \times d$ blocks gives an orthonormal basis of vectors, spanning the (filter specific) wavelet domain of $L^2(\Re)$. The eigenvectors of the covariance matrix are guaranteed to be real since the matrix is real and symmetric. The covariance matrix is symmetric since all the error vectors used in its calculation were symmetric (since they were symmetrised). Since the basis is composed solely of eigenvectors, it is orthonormal. Since the basis is orthonormal, we need not worry about cross correlations between vectors. We may now project the errors from coded blocks in each sub-band onto this orthonormal basis of d^2 blocks to eliminate the error. This obviously introduces another step into the reconstruction process. Once the affine maps have been applied

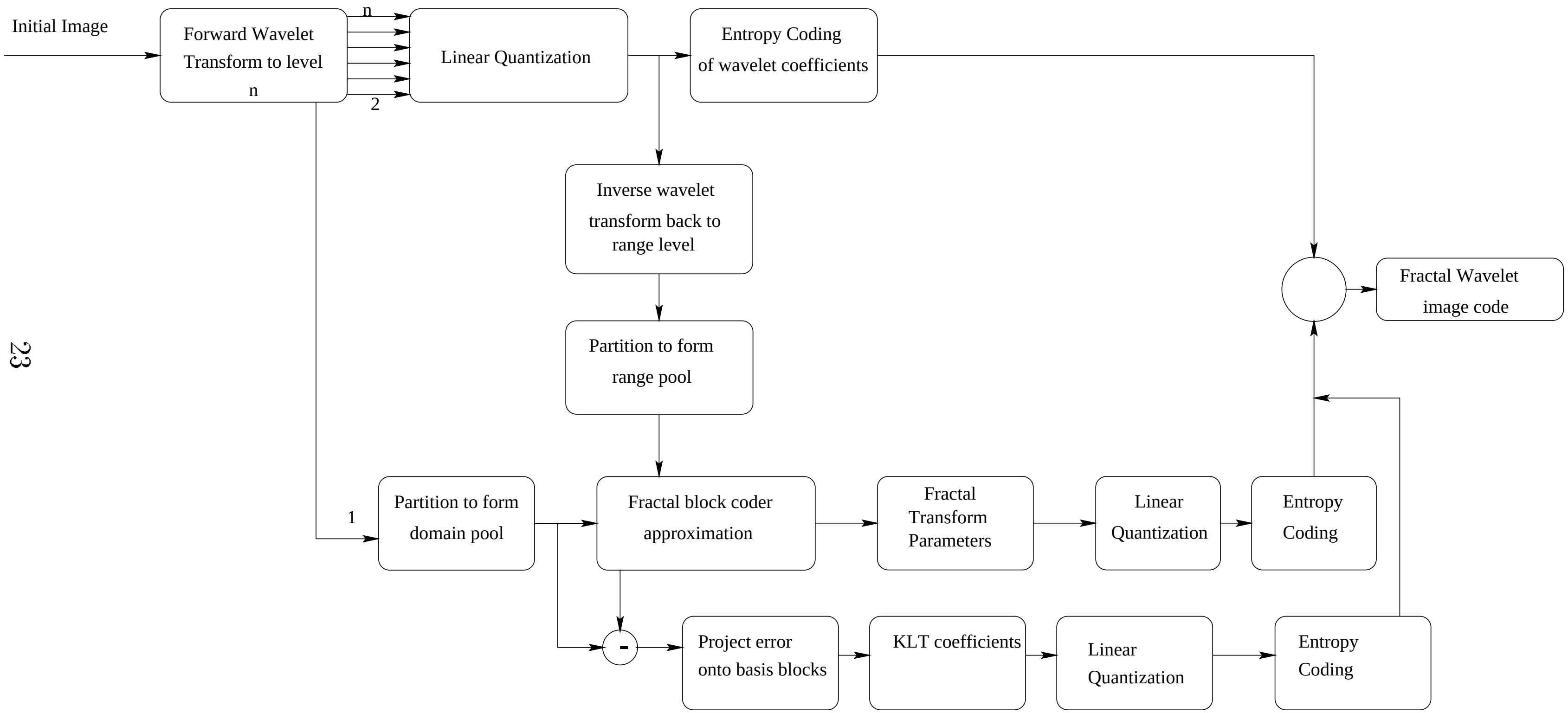


Figure 11: Modified coder block diagram

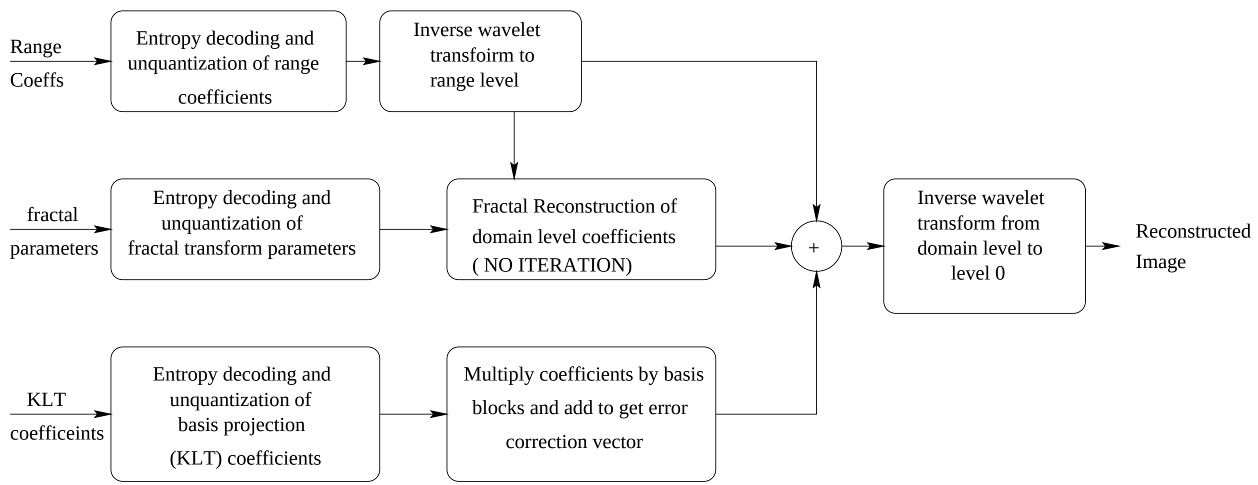


Figure 12: Modified decoder block diagram

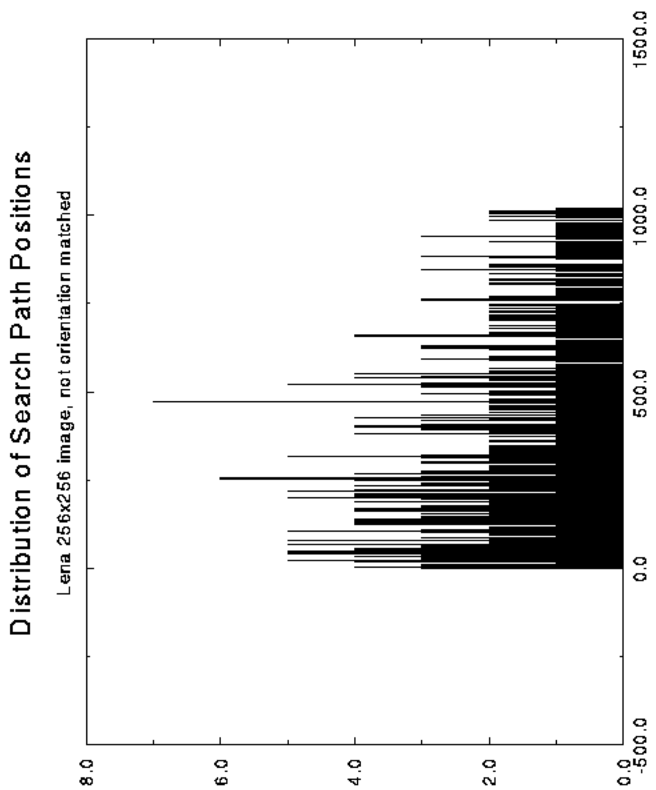


Figure 13: Distribution of path positions when block orientations are not matched

for a block, we then correct the error by adding in the basis blocks with appropriate scale factors. A block diagram of the modified decoder is shown in figure 12.

5 Results

In this section we shall provide some results of application of the new algorithm to standard test images. Comparisons with other compression systems will also be made. For comparison purposes, all test images are 512×512 pixels with 8 bit greyscale. The wavelet filter used in all tests is the Daubechies 8 point filter described in Section 3.1. As per most of the image processing literature, we shall use the *peak signal to noise ratio* (PSNR) to evaluate and compare our results. Unless otherwise stated, the orientation quantiser is set to 8 bins, the base wavelet level (n in the coder diagram) is set to 5 and the domain block size is 4×4.

5.1 Results

Table 2 shows the results of using the coder (with first range coefficient coder) with a rate value of 512 and varying the wavelet quantiser rate. Basis projection is not enabled and the coder does no searching for the best range block. That is, it simply copies the one in the same relative position from the level above. Table 3 shows the coder using the same parameters but with basis projection enabled. Figure 14 shows these results graphically.

Wavelet Rate Value	*MSE*	*PSNR*	*Bits per Pixel*
8	18.47	35.47	0.85
16	20.98	34.91	0.65
32	27.30	33.77	0.53
64	45.32	31.57	0.36
128	87.45	28.71	0.25
256	164.43	25.97	0.19

Table 2: Results for initial fractal block coding in the wavelet domain. Level 2 to Level 1, domain block size 4, no search, basis projection not enabled.

Wavelet Rate Value	*MSE*	*PSNR*	*Bits per Pixel*
8	11.49	37.53	0.93
16	14.06	36.65	0.73
32	20.64	34.98	0.57
64	38.37	32.29	0.43
128	80.22	29.09	0.34
256	157.08	26.17	0.27

Table 3: Results for initial fractal block coding in the wavelet domain. Level 2 to Level 1, domain block size 4, no search, basis projection enabled.

Table 4 the effects of enabling a 4 block radius search and figure 15 shows the rate distortion curve.

Table 5 shows the effect of the coefficient rate value on a system where the wavelet quantisation rate is held constant (at 32 in this case), no search and the basis projection enabled. Figure 17 shows the rate/distortion curve for the basic coder with no basis projection, with and without searching enabled. Table 6 shows the results of straight wavelet coding and Figure 16 shows a rate/distortion curve for this. Figure 18 and Figure 19 show the effects of varying the wavelet coefficient quantiser and the rate value with basis projection enabled.

Wavelet Rate Value	*MSE*	*Coefficient Size*	*Bits per Pixel*
8	14.17	36.62	1.01
16	17.41	35.72	0.80
32	24.37	34.26	0.63
64	43.07	31.79	0.49
128	86.50	28.76	0.38

Table 4: Results for initial fractal block coding in the wavelet domain. Level 2 to Level 1, domain block size 4, 4-block radius search, basis projection not enabled.

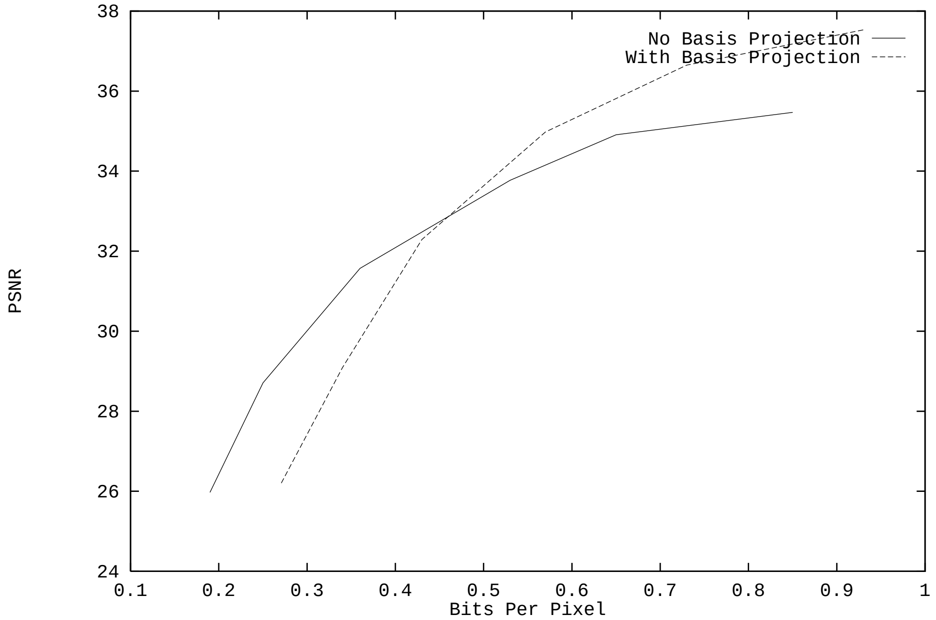


Figure 14: Comparison of reconstruction errors when the basis projection is enabled and when it isn't.

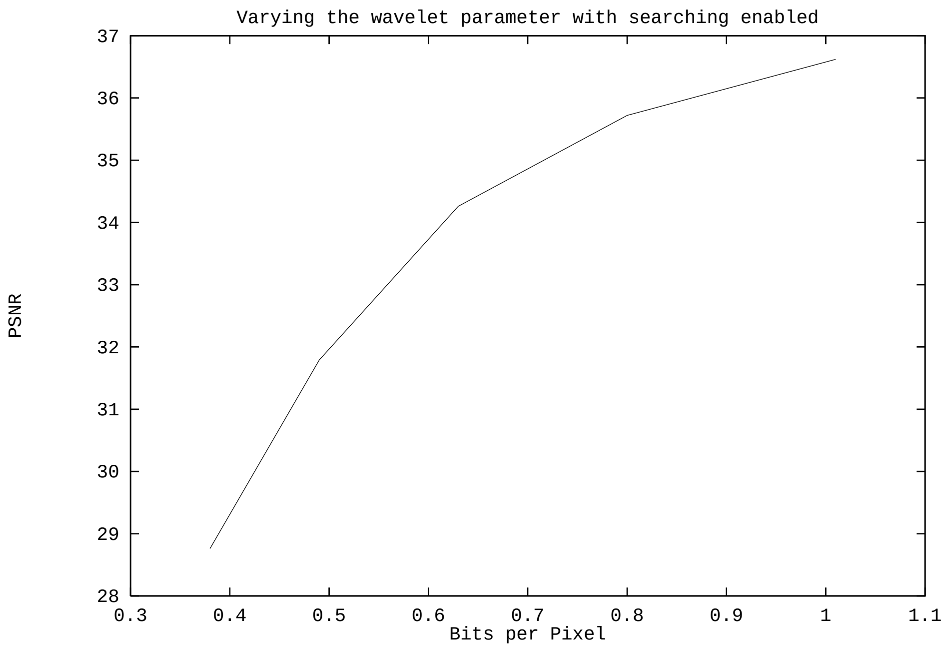


Figure 15: Rate Distortion Curve for a coder with the search enabled

Rate Value	*MSE*	*PSNR*	*Bits per Pixel*
8	10.83	37.78	4.28
32	11.07	37.69	2.70
64	11.79	37.42	1.96
128	13.82	36.72	1.26
256	16.96	35.84	0.78
512	20.64	34.98	0.57
1024	24.07	34.32	0.48
2048	26.68	33.87	0.44

Table 5: Results for initial fractal block coding in the wavelet domain. Level 2 to Level 1, domain block size 4, no search, basis projection enabled.

Wavelet Rate Value	*MSE*	*PSNR*	*Bits per Pixel*
8	5.53	40.70	1.69
16	11.33	37.59	0.92
32	21.76	34.75	0.50
64	43.71	31.72	0.26
128	88.95	28.64	0.13
256	171.63	25.78	0.06
512	309.01	23.23	0.03
1024	526.80	20.91	0.02

Table 6: Results for straight wavelet quantisation coding

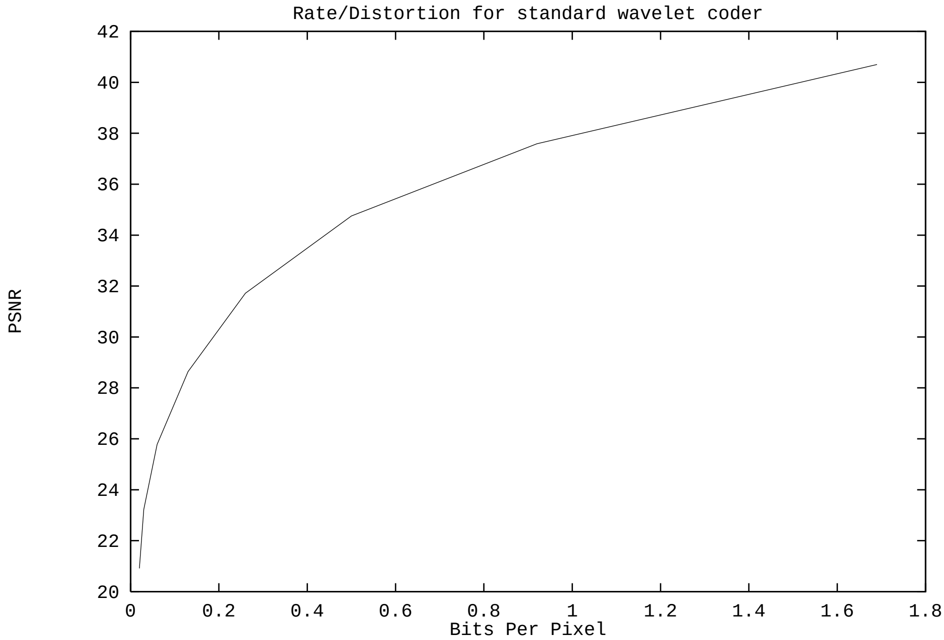


Figure 16: Rate distortion curve for straight wavelet coder

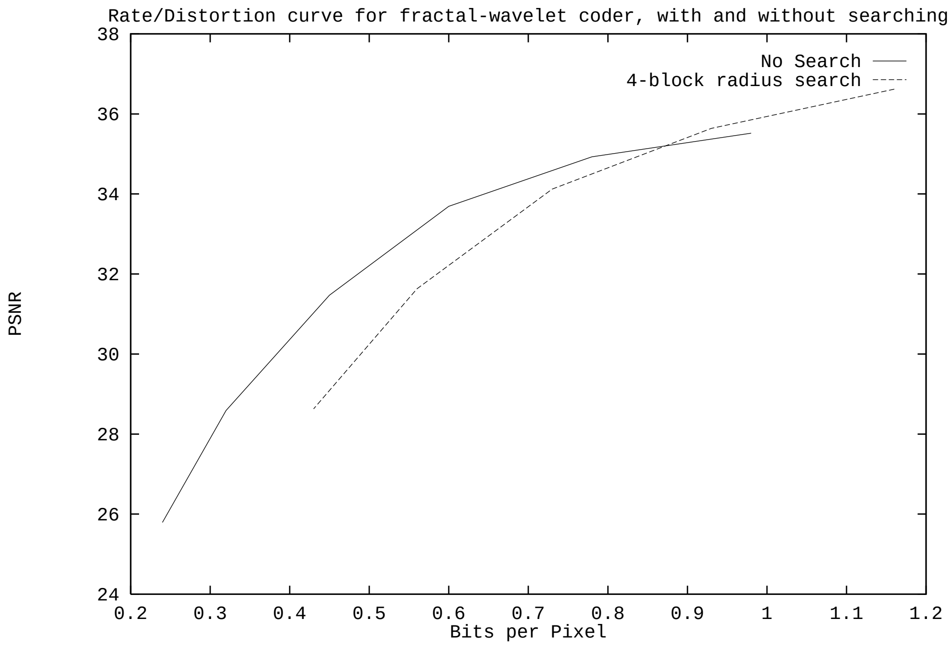


Figure 17: Comparison of basic coder, with and without searching

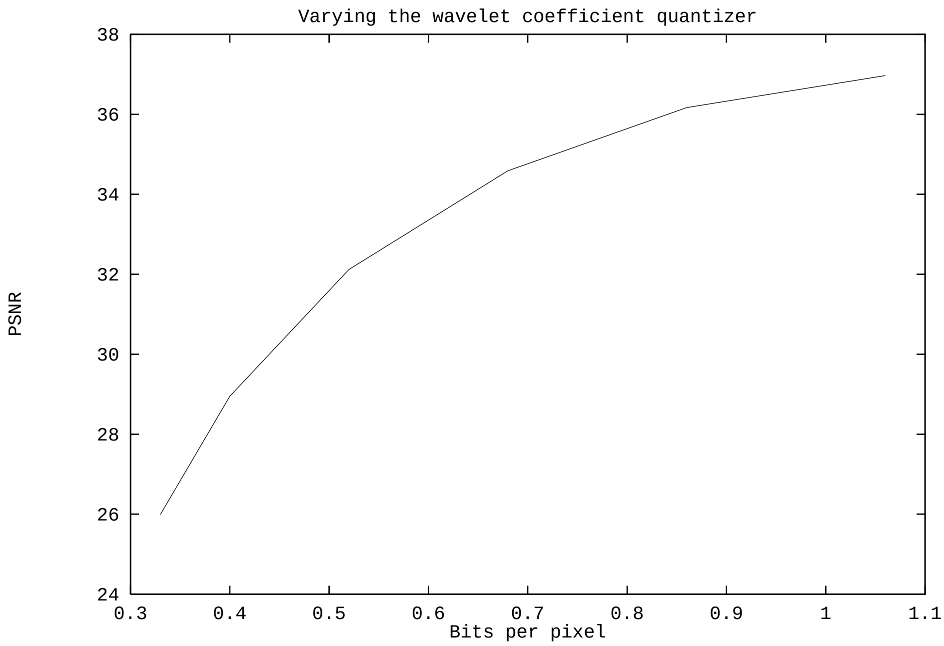


Figure 18: Varying the wavelet coefficient quantiser

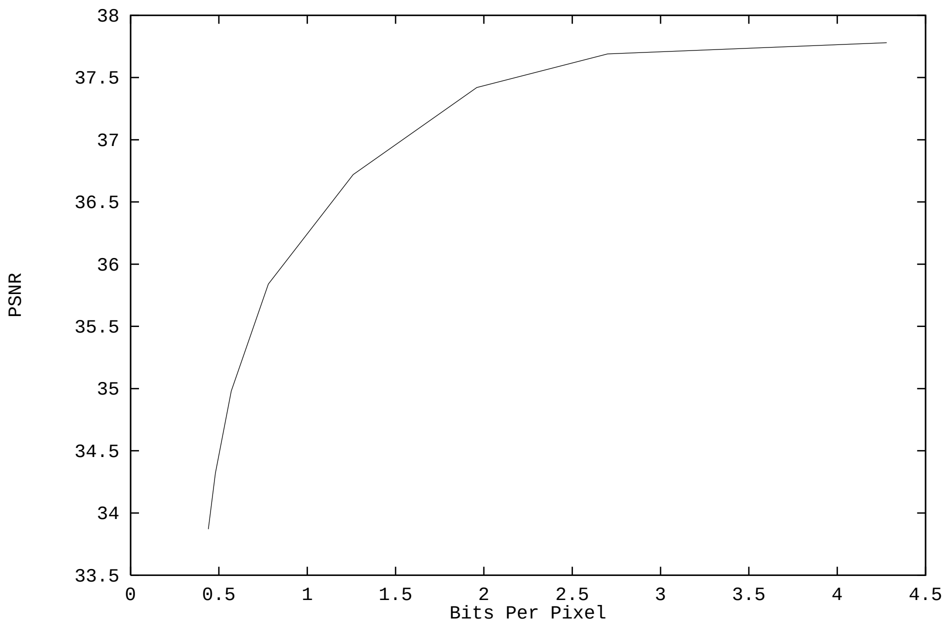


Figure 19: Varying the rate value

Figure 20: Fractal-Wavelet reconstruction at 0.35 bpp, PSNR 31.93

5.2 Test Image results

In this section, we present some reconstructed images at various bit rates and compare with other compression schemes.

Figure 20 shows the reconstructed image of Lena, 512x512 pixels, using basis projection and no search. Figure 21 shows the reconstructed image generated by quantising the range coefficients alone, at a similar error rate. For comparison, Figure 22 shows the current industry standard at approximately the same bit rate as the fractal-wavelet coder.

5.3 Modified Results

The final test for the novel coder is to change the domain block size. We provide results for 4,8 and 16 pixel square blocks at the same rate value. Table 7 shows the results of varying the domain block size. Note that no basis projection can occur currently for block sizes above 4.

The images in Figures 24 and 23 show the reconstructions from the 8 point and 16 point block sizes. For comparison, figure 25 shows the JPEG reconstruction at roughly the same bit rate. It is apparent that the JPEG system produces more visible artifacts. The novel coder provides a gain of nearly 1dB over the JPEG coder.

Figure 21: Wavelet only reconstruction at 0.26 bpp, PSNR 31.72

Figure 22: JPEG coded reconstruction at 0.33 bpp, PSNR 33.83

Domain Block Size	*MSE*	*PSNR*	*Bits per Pixel*
4	45.32	31.57	0.36
8	46.11	31.49	0.25
16	46.40	31.47	0.21

Table 7: Results for varying the domain block size. In each case, the rate value is 512 and the wavelet quantiser is set to 64. There is no searching or basis projection enabled.

Figure 23: Reconstruction using a block size of 16 pixels, 0.21bpp PSNR 31.47

Figure 24: Reconstruction using a block size of 8 pixels, 0.25 bpp, PSNR 31.49

Figure 25: JPEG Reconstruction at 0.22 bpp, PSNR 30.71

6 Discussion,Conclusions and Further Work

6.1 Discussion of Results

The final coder presented in this paper, along with its modified range coder, outperforms the current industry standard, JPEG. It provides nearly 1dB gain over JPEG at 0.21 bits per pixel. It does not, however, compete as well with other wavelet based coders, such as Shapiro's embedded zerotree algorithm [22]. This method of image coding does, however, show promise.

6.2 Summary

In this paper, we have reviewed the state of block-wise fractal image coding and presented a brief overview of wavelet analysis as it applies to this paper. We then presented a novel image coding technique that combines properties of both fractal coders and the wavelet transform, coining the name Fractal-Wavelet coding. Some test results on a standard image are then presented.

6.3 Further Research Direction

The most apparent place for further experimentation is the basis projection system. A more intelligent method of selecting basis vector coefficients and a way of propagating basis vector coefficients through levels, possibly based on eigenvalues, would also be beneficial. Further research could also yield the effect of using different mother wavelets (ie QMF filter pairs), including non-separable transforms, and including an adaptive block size system, possibly quadtree partitioning. The arithmetic coder statistical model could also be extended to be of a higher order. As an extension to this system, coding image sequences could be considered. The wavelet coefficients could be used exploit inter-frame redundancy as well as intra-frame redundancy. The basic method here would be that the reconstruction of the previous frame would act as the range coefficients for the current frame.

References

[1] M. Barnsley. *Fractals Everywhere.* Academic Press, 1988.

[2] M. Barnsley and S. Demko. Iterated Function Systems and the Global Construction of Fractals. In *Proc. Royal Society of London*, 1985.

[3] M. Barnsley and L. Hurd. *Fractal Image Compression.* AK Peters, 1992.

[4] K. Barthel and T. Voyé. Adaptive Fractal Image Coding in the Frequency Domain. In *Proceedings of International Workshop on Image Processing*, 1994.

[5] C. Chui. *An Introduction to Wavelets.* Academic Press, 1992.

[6] I. Daubechies. Orthonormal Bases of Compactly Supported Wavelets. In *Communications on Pure and Applied Mathematics*, volume XLI, 1988.

[7] J. Friedman, J.Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. In *ACM Trans. Math. Software 3,3*, 1977.

[8] M. Gharavi-Alkhansari and T. Huang. Fractal-Based Techniques for a Generalized Image Coding Method. In *Proc. IEEE ICASSP*, 1994.

[9] E. Jacobs, Y. Fisher, and R. Boss. Iterated Transform Image Compression. Technical Report 1408, Naval Ocean Systems Centre, April 1991.

[10] E. Jacobs, Y. Fisher, and R. Boss. Image Compression : A study of the iterated transform method. *Signal Processing*, 29, 1992.

[11] A. E. Jacquin. A Novel Fractal Block-Coding Technique for Digital Images. In *Proc. ICASSP '90*, 1990.

[12] G. Langdon. An Introduction to Arithmetic Coding. In *IBM Journal of Research and Development*, volume 28, 1984.

[13] S. Mallat. Multiresolution approximation and wavelet orthonormal bases of $l^2(\Re)$. In *Trans. Amer. Math Soc.*, volume 315, 1989.

[14] D. Monro. Class of Fractal Transform. In *Electronics Letters*, volume 29, 1993.

[15] D. Monro and F. Dudbridge. Fractal Approximation of image Blocks. In *Proc. IEEE ICASSP*, 1992.

[16] D. Monro, F. Dudbridge, and A. Wilson. Deterministic rendering of self-affine fractals. In *IEE Colloquium on Fractal Techniques in Image Processing*, 1990.

[17] D. Monro, D. Wilson, and J. Nicholls. High Speed Image Coding with the Bath Fractal Transform. In *Multimedia*, 1993.

[18] D. Monro and S. Woolley. Fractal Image Compression Without Searching. In *Proc. IEEE ICASSP*, 1994.

[19] W. Pennebaker and J. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1992.

[20] B. Ramamurthi and A.Gersho. Classified Vector Quantization of Images. *IEEE Trans. Commun.*, 34, Nov 1986.

[21] D. Saupe. Accelerating Fractal Image Compression by Multi-Dimensional Nearest Neighbour Search. In *Proceedings DCC'95 Data Compression Conference*, 1995.

[22] J. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. In *IEEE Transactions on Signal Processing*, volume 41, 1993.

[23] R. Williams. *Adaptive Data Compression*. Kluwer Academic Publishers, 1991.

[24] R. Wilson. BMVC93 Tutorial Notes on Wavelet Transforms. In *British Machine Vision Conference*, 1993.

[25] I. Witten, R.Neal, and J. Cleary. Arithmetic Coding for Data Compression. In *Communications of the ACM*, volume 30, 1987.