Contents

List of Figures

List of Tables

Abstract

Fractal image compression based on the use of iterated function systems (IFS), is a class of image compression techniques which offer the advantages of high compression ratio and fast decoding. In this thesis we present several new techniques to increase the efficacy of fractal image compression.

Fractal image compression requires high encoding times due to the requirement of search for the best fitting domain block for each range block. We show that matching of domain and range blocks can be more efficiently done by an inner product of range and domain blocks, than by the traditional least squares regression. We then introduce a new FFT-based block matching technique, which can considerably increase the speed of the encoding process. We also present a hybrid scheme of image compression wherein the first stage of compression is a conventional low-bit-rate fractal coding followed by a method of encoding errors by FFT-based block matching.

We introduce a new family of Feature Highlighting Transforms (FHT) which are a class of unitary transforms in which the basis vectors are chosen to highlight a desired feature of the signal. The FHT is then used to highlight the edges in images before they are encoded using a low-bit-rate fractal coding scheme. Coding in conjunction with FHT results in a much better reproduction of edges than coding without FHT. Extensions of the FHT which permit a trade-off between resolution and fidelity are also briefly explored.

We then explore the well known property of size invariance of the IFS code which permits use of IFS coding for image interpolation. It is observed that IFS codes with larger range blocks are better suited for interpolation than codes with small range blocks. However, increasing the size of the range blocks results in a reduction in the peak signal to noise ratio (PSNR). We therefore introduce a new IFS coding scheme in which multiple domain blocks are used to approximate each range block and thus an improvement in PSNR is achieved without reducing the size of the range blocks. We also study the usefulness of the FHT used in conjunction with IFS coding for interpolation of images. Finally we make a comparative study of interpolation using IFS with DCT and Subband interpolation.

Chapter 1

Introduction

Image Data Compression [1] has become an important issue for the purpose of storage and transmission of images or sequences of images, due to their large memory requirements. Most of the methods in use can be classified under the head of 'lossy' compression. This implies that the reconstructed image is always an approximation of the original image. Thus to be effective, one should take the human visual system into account before designing the compression scheme.

The most popular among these methods is the discrete cosine transform(DCT) [2], which is known to be the closest to the ideal energy compaction transform, the Karhunen-Louve Transform (KLT), for images [1]. The DCT is also the JPEG standard [3] for still image compression. However in the recent past some new methods have been investigated which perform as well or better than the DCT in most cases. Notable among them are the wavelet transform, [4, 5] (which is closely related to subband coding [6] and pyramid decomposition [7]), and iterated function systems (IFS) image compression [8, 9, 10, 11], more commonly referred to as *fractal* image compression.

IFS image compression was first suggested by Barnsley [11] in 1988. However the first automated compression scheme for real world images using IFS was developed by Jacquin [8, 9] in 1990. During the last six years immense research activity has taken place in this field. As a result, currently, IFS image compression is considered to be comparable to the existing methods at high and moderate bit rates (.5 to 1 bpp) and superior to most methods at low bit rates (< 0.25 bpp). The main disadvantage of the IFS scheme is that the encoding process is computationally very intensive. However decoding is simple and fast. This method is thus ideally suited for browsing archives where encoding is done only once, while decoding is done often.

1.1 Mathematical Foundation

There are four main mathematical concepts [12] underlying IFS image compression: metric spaces, contractive maps, the contractive mapping fixed point theorem, and the collage theorem.

Definition 1.1 : *A metric space $(\mathcal{X}, d)$ is a set $\mathcal{X}$ with a real-valued distance measure (metric) $d : \mathcal{X} \times \mathcal{X} \to \Re$, defined with the following properties:*

1. $d(a,b) \geq 0 \ \forall \ a,b \ \in \mathcal{X}$.
2. $d(a,b) = 0$ *iff* $a = b \ \forall \ a,b \ \in \mathcal{X}$.
3. $d(a,b) = d(b,a) \ \forall \ a,b \ \in \mathcal{X}$.
4. $d(a,c) \leq d(a,b) + d(b,c) \ \forall \ a,b,c \ \in \mathcal{X}$. (*Triangle Inequality*)

In most of our applications the metric space $(\mathcal{X}, d)$ is the space of $M \times N$ matrices whose elements correspond to pixel values of images. For the metric d we use the squared error. For example if $\boldsymbol{A} = \langle a_{ij} \rangle, \boldsymbol{B} = \langle b_{ij} \rangle \in \mathcal{X}$ then

$$d(\boldsymbol{A}, \boldsymbol{B}) = \sum_{i=1}^{M} \sum_{j=1}^{N} (a_{ij} - b_{ij})^2. \tag{1.1}$$

Definition 1.2: *A sequence of points $\{x_n\}$ in a metric space is called a* **Cauchy sequence** *if for every $\epsilon > 0$ there exists an integer N such that*

$$d(x_m, x_n) < \epsilon \ \forall \ n,m > N. \tag{1.2}$$

Definition 1.3: *A metric space $\mathcal{X}$ is* **complete** *if every Cauchy sequence in $\mathcal{X}$ converges to a limit point in $\mathcal{X}$.*

Definition 1.4: *Let $(\mathcal{X}, d)$ be a metric space. A map $w : \mathcal{X} \to \mathcal{X}$ is* **Lipschitz** *with Lipschitz factor s if there exists a positive real number s such that*

$$d(w(\boldsymbol{A}), w(\boldsymbol{B})) \leq \ sd(\boldsymbol{A}, \boldsymbol{B}) \ \forall \ \boldsymbol{A}, \boldsymbol{B} \ \in \mathcal{X}. \tag{1.3}$$

If the Lipschitz factor $s < 1$, then w is said to be **contractive** *with* **contractivity** *s.*

Theorem 1.1: **(The Contraction Mapping Fixed Point Theorem)** *Let $\mathcal{X}$ be a* *complete metric space and $w : \mathcal{X} \to \mathcal{X}$ be a contractive mapping. Then there exists a unique fixed point $\boldsymbol{F} \in \mathcal{X}$ such that for any point $\boldsymbol{P} \in \mathcal{X}$*

$$\boldsymbol{F} = w(\boldsymbol{F}) = \lim_{n \to \infty} w^{\circ n}(\boldsymbol{P}). \tag{1.4}$$

Such a point is called a **fixed point** *or the* **attractor** *of w.*

Definition 1.5: *A collection $w_1, w_2, \cdots, w_n$ of contractive maps on a metric space $(\mathcal{X}, d)$ is called iterated function system* (IFS).

Armed with the above mathematical concepts we are ready to investigate how IFS can be used to encode images. Consider an image matrix $\boldsymbol{A} \in \mathcal{X}$. Further let

$$\hat{\boldsymbol{A}} = w(\hat{\boldsymbol{A}}) = \lim_{n \to \infty} w^{\circ n}(\boldsymbol{P}) \ \forall \ \boldsymbol{P} \in \boldsymbol{X}, \tag{1.5}$$

where w is a contractive transformation, and $d(\boldsymbol{A}, \hat{\boldsymbol{A}})$ is ‘small’. Thus if w can be determined and represented by fewer bits than $\boldsymbol{A}$ we have achieved compression. This is due to the fact that we can arrive at $\hat{\boldsymbol{A}}$ iteratively by applying w to any arbitrary point (image) $P \in \mathcal{X}$.

Unfortunately, these theorems do not provide us with a method of arriving at the map w. This problem is simplified to a large extent by the collage theorem.

Theorem 1.2 (**Collage Theorem**): *Let $(\mathcal{X}, d)$ be a metric space and $w : \mathcal{X} \to \mathcal{X}$ be a contractive mapping with contractivity s. Let $\hat{\boldsymbol{A}}$ be the fixed point of w. Then*

$$d(\boldsymbol{A}, \hat{\boldsymbol{A}}) \leq \frac{1}{1-s} d(\boldsymbol{A}, w(\boldsymbol{A})). \tag{1.6}$$

What the collage theorem implies is the following. Let $\boldsymbol{A} \in \mathcal{X}$ be an image. Let $w : \mathcal{X} \to \mathcal{X}$ be such that $d(w(\boldsymbol{A}), \boldsymbol{A}) < \epsilon$. Then $d(\hat{\boldsymbol{A}}, \boldsymbol{A})$ is also bounded. Therefore the problem of finding the fractal code w for an image $\boldsymbol{A}$ can be restated as follows:

Find w such that $d(\boldsymbol{A}, w(\boldsymbol{A}))$ is ‘small’.

Usually w is of the form

$$w = \bigcup_{i=1}^{N} w_i, \tag{1.7}$$

where each w_i is a contractive mapping. However, this is not strictly required. It is sufficient if w is *eventually contractive.*

Definition 1.6: *Let $(\mathcal{X}, d)$ be a metric space and $w : \mathcal{X} \to \mathcal{X}$ be a Lipschitz function such that $w^{\circ n}$ is contractive, then w is* **eventually contractive**. *It follows that for any* $\boldsymbol{P} \in \boldsymbol{X}$

$$\hat{\boldsymbol{A}} = w(\hat{\boldsymbol{A}}) = \lim_{k\to\infty} w^{\circ k}(\boldsymbol{P}). \tag{1.8}$$

The integer n is called the exponent of eventual contractivity.

For eventually contractive mappings, the collage theorem is slightly modified.

Theorem 1.3 (**Collage Theorem for Eventually Contractive Mappings**): *Let w be an eventually contractive mapping with exponent n. Let $\hat{\boldsymbol{A}}$ be the fixed point of w. Then*

$$d(\boldsymbol{A}, \hat{\boldsymbol{A}}) = \frac{1}{1-s}\frac{1-\sigma^n}{1-\sigma} d(\boldsymbol{A}, w(\boldsymbol{A})), \tag{1.9}$$

where s is the contractivity of $w^{\circ n}$ and σ is the Lipschitz factor of w [12].

Now we will see how the collage theorem helps in simplifying the job of finding the mapping w for an arbitrary image. Let us assume, without any loss of generality, that the region of support of the image is the unit square $\mathcal{I}^2 = [0,1] \times [0,1]$ and the dynamic range of the grey level of the image is scaled to the interval $\mathcal{I} = [0,1]$. The parametric form of the transformations $w_i : \mathcal{I}^3 \to \mathcal{I}^3, i = 1 \cdots N_r$ is chosen as

$$w_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}, \tag{1.10}$$

where (x, y) denotes the coordinates of a point in $\mathcal{I}^2$ and $z = g(x, y)$ denotes the intensity or grey level at (x, y). Each transformation w_i can be split into two transformations $L_i : \mathcal{I}^2 \to \mathcal{I}^2$ and $T_i : \mathcal{I} \to \mathcal{I}$, defined as

$$L_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}, \tag{1.11}$$

and

$$T_i(z) = s_i z + o_i. \tag{1.12}$$

Associated with each w_i is a range block $\boldsymbol{R}_i \in \mathcal{I}^2$ and a domain block $\boldsymbol{D}_i \in \mathcal{I}^2$. The N_r range blocks which may or may not be of equal size, are non-overlapping and tile the unit square $\mathcal{I}^2$ completely. The transformations L_i and T_i and the domain blocks $\boldsymbol{D}_i$ are chosen to satisfy the following conditions:

- L_i maps $\boldsymbol{D}_i$ into $\boldsymbol{R}_i$,
- the size of each $\boldsymbol{D}_i$ is larger than that of the corresponding $\boldsymbol{R}_i$ to ensure contractivity, and,
- The collage error, given by

$$\varepsilon_i = \int_{\boldsymbol{R}_i}\int [g(x,y) - T_i(g(L_i^{-1}(x,y)))]^2 dxdy \tag{1.13}$$

  is minimum.

Additionally, the condition $s_i < 1$ must be satisfied for contractivity, but it is not a necessary condition for eventual contractivity. The parameters a_i, b_i, c_i and d_i, determine the spatial contraction factor and the isometric operation performed on the block, *i.e.*, rearrangement of pixels within a domain block. The parameters e_i and f_i give the physical distance or the address of the domain block with reference to the range block. The parameters s_i and o_i are respectively the intensity scaling factor and the offset.

Note that in (1.13) we call upon the collage theorem to optimize the parameters of the mapping for each w_i separately. Without the collage theorem we would have to do a simultaneous optimization for all the mappings, which would be impractical. Though the w arrived at through the collage theorem is suboptimal, the collage theorem is what has made IFS image compression a practical scheme.

The parameters of the transformation for each range block $\boldsymbol{R}_i$ become the IFS code for the image. If the IFS is contractive, and if it is applied iteratively on any arbitrary initial image (say an image of all zeros), the image converges to the fixed point, which is an approximation of the encoded image. It should be noted here that the IFS are constituted by *affine* transformations. (If the transformations are linear then an image of all zeros is a fixed point and can be the *only* fixed point).

The decoded image

$$\hat{\boldsymbol{A}} = \lim_{n\to\infty} w^{\circ n}(P) \tag{1.14}$$

is guaranteed to be 'close' to the original image if ε is 'small', (by the collage theorem), where $\varepsilon = d(\boldsymbol{A}, w(\boldsymbol{A}))$. In practice, for images, convergence is achieved in 6 to 8 iterations. The 'closeness' of the reconstructed image to the original image is usually represented by the peak signal to noise ratio (PSNR) of the decoded image. The PSNR of $\hat{\boldsymbol{A}}$, the approximation of $\boldsymbol{A}$, is given by

$$\mathrm{PSNR} = 10 \log_{10} \frac{(2^b - 1)^2 MN}{d(\boldsymbol{A}, \hat{\boldsymbol{A}})} \tag{1.15}$$

where b is the number of bits used to code the pixel values of A and the metric d is the squared error defined in (1.1).

1.2 Organization of the Thesis

The remainder of the thesis is organized as follows: The second Chapter is a review of the current state of the art of IFS or fractal image compression. In the third Chapter we propose a method to reduce the encoding complexity of the IFS scheme. In this method, least squares regression, which is normally used for determining the scaling parameter s is replaced by a simpler scalar product of range and domain blocks. In Chapter 4 we introduce a FFT-based block matching procedure which further speeds up the block matching process. We make use of the fact that correlation of two vectors of length N has a computational complexity of N^2, but if implemented through the FFT has a complexity of the order of $N \log_2 N$. We therefore use all possible circular shifts of a domain block (after spatial contraction) as the permitted isometric transformations. Still more reduction is encoding complexity can be achieved if we make a suboptimal choice of the domain blocks by comparing the magnitude Fourier coefficients before correlation is attempted. In Chapter 4 we also propose a hybrid scheme where the image is first coded by a low-bit-rate fractal compression scheme, and the FFT-based block matching technique is used for coding residual errors. In Chapter 5 we introduce a new class of unitary, feature highlighting transforms (FHT) which have impulse responses of all-pass filters as basis vectors. The FHT is used to pre-emphasize the edges before the image is coded using IFS. Upon reconstruction the inverse FHT is applied. We show that images coded in this way have significantly better reproduction of edges than the images that are coded using IFS alone. Other variants of the FHT are also briefly explored. Chapter 6 is a study of interpolation of images using the IFS code. We show how choice of different range block sizes affect the quality of interpolation. We introduce a new form of IFS where each range block is coded with multiple domain blocks. We also compare IFS interpolation with DCT and subband interpolation. A summary of results and conclusions is presented in Chapter 7.

Chapter 2

Review of IFS Image Coding

In this Chapter we review the state of the art techniques in IFS image compression.

2.1 Jacquin's method

The method proposed by Jacquin in 1990 [8, 9] and subsequently refined by Fisher [13, 14, 15] is as follows. Unlike the method described in Chapter 1, not all possible domain blocks are searched. The domain pool is restricted to $[\boldsymbol{D}_k], k = 1 \cdots N_d$. The spatial contraction factor is also fixed at 2. And only eight isometric operations are permitted (four reflections and four rotations). The isometric operations thus in effect expand the pool of domain blocks by eight. The expanded pool of domain blocks is also called the codebook library (following the terminology used in vector quantization of images). The mapping of a domain block to a range block is illustrated in Figure 2.1.

With these restrictions it becomes simpler to ignore the matrix form of the IFS given by (1.11) and consider the following scenario:

- The image is discretized to an $M \times N$ matrix of pixels. For simplicity, we will assume $M = N$.

- The image is segmented into N_r non-overlapping range blocks of size $r \times r$ and the range blocks tile the entire image. The range blocks are denoted by $\boldsymbol{R}_i, i = 1 \cdots N_r$.

- The domain pool consists of N_d domain blocks of size $d \times d$. We will further assume that $d = 2r$. The domain blocks are denoted by $\boldsymbol{D}_k, k = 1 \cdots N_d$.

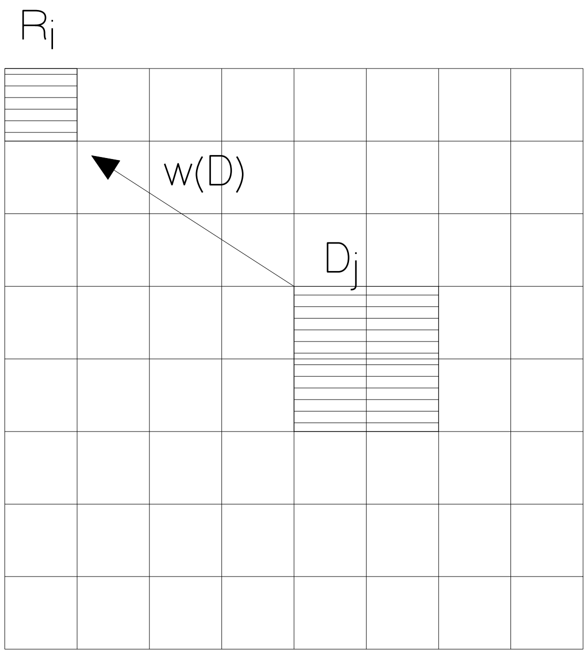

Figure 2.1: Mapping of domain to range blocks.

- S is a spatially contractive operator, which operates on domain blocks of $d \times d$ pixels to yield blocks of $r \times r$ pixels. S may be a simple sub-sampler, which means that pixel values in each transformed block are assigned by sub-sampling the corresponding domain block. More usually, the pixel values of 4 adjacent pixels in a domain block are averaged to obtain the value of the corresponding pixel in the transformed block. Pixel values in the transformed block may also be obtained by low-pass filtering followed by sub-sampling.

- The transformations T, are a set of t isometric operations (re-arranging of pixels within a block) performed on the spatially contracted domain blocks.

Thus each domain block is spatially contracted and acted upon by the transformations T resulting in t codebook blocks per domain block. The codebook blocks are denoted by $\boldsymbol{C}_j, j = 1 \cdots N_d t$.

$$\boldsymbol{C}_j = T_q(S(D_k)),\ j = 1 \cdots N_d t; k = 1 \cdots N_d; q = 1 \cdots t. \tag{2.1}$$

Now the objective is to find a codebook block $\boldsymbol{C}_j$ from the available pool, for each $\boldsymbol{R}_i$, such that the error

$$\varepsilon = d(\boldsymbol{R}_i, s_i\boldsymbol{C}_j + o_i) = \sum_{l=1}^{p}(s_i c_{jl} + o_i - r_{il})^2 \tag{2.2}$$

is minimum. In (2.2), $p = r^2$ is the number of pixels in each range block, and r_{il} and c_{jl} are the values of the l^{th} pixels in $\boldsymbol{R}_i$ and $\boldsymbol{C}_j$ respectively. The values s_i and o_i are found using least squares regression (LSR) [16] as follows. Setting the partial derivatives of ε, viz., $\frac{\partial \varepsilon}{\partial s}$ and $\frac{\partial \varepsilon}{\partial o}$, to zero, we get the error minimizing values of s and o as

$$s_i^j = \frac{p(\sum_{l=1}^{p} c_{jl}r_{il}) - (\sum_{l=1}^{p} c_{jl})(\sum_{l=1}^{p} r_{il})}{p\sum_{l=1}^{p} c_{jl}^2 - (\sum_{l=1}^{p} c_{jl})^2}, \tag{2.3}$$

and

$$o_i^j = \frac{1}{p}\Big(\sum_{l=1}^{p} r_{il} - s_i^j \sum_{l=1}^{p} c_{lj}\Big). \tag{2.4}$$

Hence the minimum error for a given (i,j) pair is given by

$$\varepsilon_i^j = \sum_{l=1}^{p}(s_i^j c_{jl} + o_i^j - r_{il})^2; i = 1, \cdots, N_r; j = 1, \cdots, N_d t. \tag{2.5}$$

For each range block $\boldsymbol{R}_i$, the codebook block $\boldsymbol{C}_j$ that gives the least error is chosen:

$$a_i = \arg\min_j \varepsilon_i^j, \tag{2.6}$$

where a_i is the address of the codebook block. Now

$$s_i = s_i^{a_i}, o_i = o_i^{a_i} \tag{2.7}$$

are the corresponding scaling and offset parameters. The values of a_i, s_i, and o_i form the fractal/IFS code for the range block $\boldsymbol{R}_i$

2.1.1 Complexity of the coder

The complexity of the coder described above can be seen with an example. Consider a 256×256 image segmented into range blocks of size 8×8 and domain blocks of size 16×16. This would result in 1024 range blocks and typically 961 domain blocks (The domain blocks are chosen to tile the entire image with each domain block overlapping half of each adjacent domain block), or 7688 codebook blocks. So the coding involves 1024×7688 LSRs of blocks of 64 pixels. Each LSR, best implemented, (as per (2.3) to (2.5)) requires approximately $2N = 2 \times 64$ multiplications. Thus the total complexity of the encoder is roughly $1024 \times 7688 \times 128$ multiplications.

However decoding complexity is linear in the number of pixels. We start with any arbitrary image and each range block $\boldsymbol{R}_i$ is replaced by $s_i \boldsymbol{C}_{a_i} + o_i$ during each iteration. The maximum value of s_i is restricted to s_{max} (Normally 1.2 to 1.5). Convergence occurs in 6 to 8 iterations.

2.2 Higher Order IFS

A more general IFS of the form

$$w = \bigcup_{i=1}^{n} w_i \tag{2.8}$$

where

$$w_i(A) = w_i \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix} = a_i x + b_i y + s_i f(x,y) + o_i \tag{2.9}$$

and $f(x,y)$ is the grey scale function of the block that is coded, was suggested by Munro and Dudbridge [17, 18]. In Jacquin's scheme the grey scale function of a block, given by (1.12), is only a function of the grey scale of the corresponding location of the domain block. It is therefore an affine transform of order zero (polynomial of degree zero in x and y co-ordinates). However, in (2.9) the grey scale function is also a function of x and y co-ordinates, which makes (2.9) an IFS of order one.

In the implementation carried out by Munro and Dudbridge, n is chosen as 4.

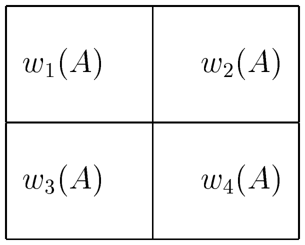

Figure 2.2: Partitioning of a square A by the IFS w_1,w_2,w_3 and w_4.

The function w_i $(i = 1, 2, 3, 4)$ maps a square A to its i^{th} quadrant as shown in Figure 2.2. The image is partitioned into non-overlapping domain blocks of size $2r \times 2r$, and range blocks of size $r \times r$. Each domain block is mapped on to the 4 range blocks contained within itself by the transformations $w_i, i = 1, 2, 3, 4$ (Compare this to Jacquin's method where the domain block corresponding to a range block may be located anywhere in the image). Hence this method generates a *self affine system*. No search is involved in the determination of the best domain block. Also, each square block is coded independently of other blocks. Each square block is coded with sixteen parameters (4 values - a_i, b_i, s_i, and o_i for each w_i or each quadrant). The parameters are obtained by solving a set of 16 linear equations

$$\frac{\partial d(A, \hat{A})}{\partial a_k} = 0;\ k = 1 \cdots 4 \tag{2.10}$$

and 12 other similar equations for b_k,s_k, and o_k, $k = 1 \cdots 4$

Later extensions of IFS coding include higher order transforms with partial search for domain blocks and isometric transformations, the Bath fractal transforms (BFT) [19, 20]. In a recent paper [21] Woolley and Munro have compared the performance of various types of the BFT and found that the best trade-off (between compression ratio and PSNR of decoded image) is achieved if second order polynomials with no search is used. For increasing compression ratio, the block sizes can be increased. Zhang et al, [22] modified the BFT by replacing $f(x, y)$ in (2.9) by mean subtracted $f(x, y)$. Though this does not modify the fixed point of the IFS, it results in a smaller spread in the values of the scale factor, thereby resulting in a slightly higher compression ratio. Moreover, decoder convergence is faster than in the original method.

2.3 Inner Product Space Approach

A fractal coder based on the inner product space approach was proposed by Oien et al [23] in 1991. In this method, the transformation used to approximate the i^{th} range block is

$$\hat{\boldsymbol{R}}_i = s_o^i T(\boldsymbol{D}) + \sum_{k=1}^{n_b} s_k^i \boldsymbol{B}_k. \tag{2.11}$$

Here $\boldsymbol{B}_k, k = 1 \cdots n_b$ are a set of orthonormal fixed basis blocks (FBB). The coefficients $s_k^i, k = 1 \cdots n_b$, are the projections of the i^{th} range block $\boldsymbol{R}_i$ on the FBBs. The coefficient s_0^i is the projection of the component of $\boldsymbol{R}_i$ that is orthogonal to all the FBB's onto the transformed domain block $T(\boldsymbol{D})$. The choice of the set of FBB's is arbitrary. Many different choices of FBB's have been proposed and explored by Vines et al [24]. Much of the 'smooth' information in the range blocks is covered by the FBBs, and the domain blocks supply the finer information. Therefore highly decimated domain blocks (8 to 16 times) are used. Jacquin's method can be considered to be a special case of this method with only one FBB whose elements are equal to one another.

2.4 Complexity Reduction Methods in Fractal Image Compression

Due to the unacceptably high complexity of encoding of the fractal image compression scheme, a lot of work has been directed at reducing the encoding complexity [25]. The methods suggested may be classified as

1. Domain Pool Restriction,
2. Block Classification,
3. Adaptive Clustering,
4. 1-D Functional Methods,
5. Feature Vectors, and
6. Transform Domain Block Matching.

2.4.1 Domain Pool Restriction

The first step in the reduction of computational complexity of the encoding process is to restrict the domain pool. Not all possible domain blocks are searched. Fisher [13] suggested restriction of the domain pool to blocks twice the size of the range blocks, with adjacent domain blocks overlapping one another by the size of the range blocks. Saupe [26] suggested restriction of domain blocks to those with high variances. Barthel et al [27] restricted the domain pool to the nearest neighbors, and search for the appropriate domain block was carried out in a spiral path around the range block.

2.4.2 Block Classification

Jacquin [8, 9] used a classification scheme similar to the one proposed by Ramamoorthi and Gersho [28] for vector quantization (VQ) of images. In this method, the domain and range blocks are classified as shade, edge and mixed blocks. Edge blocks are further classified as simple and mixed edge blocks. For a range block from any one of the above categories, only domain blocks from the same category are searched.

A more elaborate classification scheme was proposed by Fisher et al [13]. If $A_i, i = 1 \cdots 4$ are the mean pixel intensities of the four quadrants of a range or domain block, then we can always orient the block (by flipping or rotation) so that the block falls into one of the following three major classes:

- Major Class 1: $A_1 \geq A_2 \geq A_3 \geq A_4$
- Major Class 2: $A_1 \geq A_2 \geq A_4 \geq A_3$
- Major Class 3: $A_1 \geq A_4 \geq A_2 \geq A_3$

The ordering of brightness of the quadrants for the three major classes are illustrated in Figure 2.3. Further a block belonging to one particular major class can be classified into $4! = 24$ ways depending on the ordering of the variances of each quadrant. Thus there are effectively 72 classes of blocks. For a range block belonging to one particular class, only domain blocks of the same class are searched. (If the scale factor can take negative values then two classes have to be searched).

Boss and Jacobs [29] developed an archetype classification scheme. In this scheme, the archetype form for a particular codebook block is given by the block that can best cover all others having the same archetype, best in the least squares sense. Starting

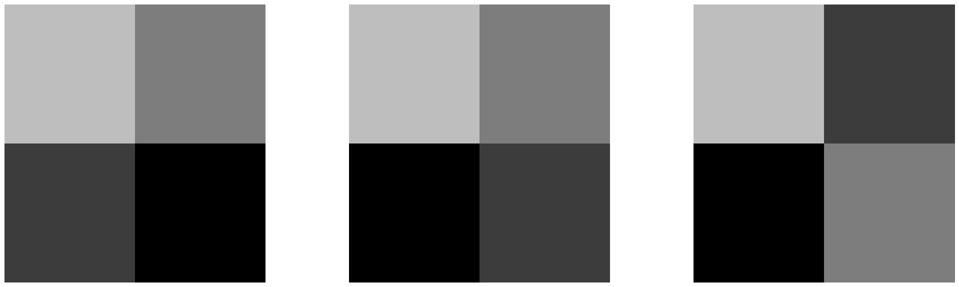

Figure 2.3: The 3 major classes of Fisher's classification scheme.

from an arbitrary classification using codebook blocks from a library made from many images, the classification scheme is iterated till the best archetype form for each block is determined. The iteration is stopped when there is no more change in the selection of archetype blocks in further iterations. These archetype blocks, derived from training images are used for classification of the range and domain blocks of images.

2.4.3 Adaptive Clustering

In the three methods described earlier, the classification scheme is decided upon before it is applied to any image. On the other hand in the adaptive clustering scheme proposed by Oien and Lepsoy, [30], the classification is image dependent. In this method, the range and codebook blocks (after removal of the FBB components from them) are divided into disjoint sets around 'cluster centers'. The cluster centers are iteratively updated if any block does not fit into any particular cluster, at the same time satisfying some conditions.

2.4.4 1-D Functional Methods

Bedford et al [31] proposed a method in which the range and domain blocks are compared by projecting each of them on a common, fixed unit vector. A particular range block is compared only with domain blocks whose result of comparison with the unit vector is close to that of the result of comparison of the range block. This method has further been extended by [25] to include more such vectors.

2.4.5 Feature Vectors

This method was first proposed by Saupe [26]. In this method, a small set of d real-valued keys are assigned to each range and domain block, which make up the d-dimensional *feature vector*. These keys are constructed such that searching in the domain pool is restricted to a small neighborhood around the d-dimensional key corresponding to a particular range block. Thus the sequential search of the domain block is substituted by a nearest neighbor search.

Kominek [32] used a much simpler scheme for arriving at the feature vector, but used the r-trees algorithm for searching. In this algorithm the d-dimensional space of the feature vectors is divided into d-dimensional rectangles (the 'r' in r-trees stand for 'rectangle'). For a range block with its feature vector located in one rectangle, only domain blocks with feature vectors in the same rectangle are searched.

Frigaard et al [33] used two dimensional feature vectors- the grey scale standard deviation (a continuous feature) and the number of dominant grey levels (a discrete feature) in each block. Bani-Eqbal [34] used 4-dimensional feature vectors.

2.4.6 Transform Domain Block Matching

Wohlberg et al [35] proposed a scheme where block matching is done in the DCT domain. Here they make use of the well known energy compaction property of the DCT to reduce the number of dimensions of the feature vector space, where a nearest neighbor search is performed. Also the fact that the DCT of the transformed blocks (for the usual transformations employed) can be obtained from the DCT of the original block with just multiplication by ± 1 is put to use. In Chapter 4 we introduce a FFT-based block matching procedure in combination with magnitude Fourier domain block matching, which can substantially speed up the block matching process.

2.5 Issues in Decoding

There are two major issues concerning decoding of the IFS code.

- Is the IFS contractive? Or in other words, will it reach the fixed point?
- In how many iterations will it reach arbitrarily close to the fixed point?

One method of ensuring contractivity of the IFS is to restrict scale factors to be strictly less than 1. But this adversely affects the quality of the approximation and hence the reconstructed image [13, 36]. Though it has always been found that the IFS contracts every time even when scale factors as high as 2 are chosen [13], one cannot guarantee convergence. Lundheim [37] has given a good mathematical treatment for finding the Lipschitz factor (which should be less than 1 for convergence) of an IFS for eventual contractivity. If $\mathcal{D}$ is the set of all domain blocks, ∇_D the set of all range blocks whose IFS code depend on the domain block D with $s_i, i \in \nabla_D$ as scale factors and r the ratio of size of the domain and range blocks, then for decimation of domain block by subsampling, the IFS is eventually contractive if

$$\max_{D\in\mathcal{D}} \sum_{i\in\nabla_D} (s_i)^2 < r. \tag{2.12}$$

For the case of decimation by averaging the IFS is eventually contractive if

$$\max_{D\in\mathcal{D}} \frac{1}{d} \sum_{i\in\nabla_D} (s_i)^2 < r, \tag{2.13}$$

where $\frac{1}{d}$ is the averaging filter weights.

Hurtgen et al [38] studied the convergence of fractal transforms by looking at the spectral radius of the linear part of the transformation matrix (the other part being the offset). However, like Lundhiem's observations, the condition for contractivity derived by Hurtgen, was only a sufficient condition (and not a necessary condition).

Kominek [36] and Domascincz et al [39] independently developed a new method for determining the convergence of the IFS code. They used the technique of mapping graphs, where the links between domain blocks are plotted in a graph, which has all the domain blocks as its nodes. The nodes that form closed loops are identified and studied individually for contraction as they are independent of other loops.

Perhaps one of the most useful contributions came from Oien and Lepsoy [40], where they show that a slight modification of the usual IFS used, viz,

$$\hat{\boldsymbol{R}} = s\boldsymbol{C} + o \tag{2.14}$$

to

$$\hat{\boldsymbol{R}} = s(\boldsymbol{C} - m_c) + m_r \tag{2.15}$$

where m_c is the mean of the codebook block and m_r is the mean of the range block, assures contraction of the IFS in a specific number of iterations, irrespective of the value of the scale factor. It can be easily verified that both (2.14) and (2.15) have the same fixed point. Convergence is reached in a specific number of iterations depending on the size of the range block and the decimation ratio. Further the decoding can be done in a hierarchical manner - the first stage of decoding is done on an image in which each pixel represents a range block, the next iteration on an image four times that size and so on. The conditions to be satisfied for this method to work are:

- Decimation should be by averaging.
- The domain blocks should be made up of an integral square number of range blocks. Therefore, if the domain block edges are twice that of the range block, we can choose domain blocks which do not overlap, or overlap by the size of range block. The latter is the normal choice.
- Though not mentioned by Oien et al in [40] the isometric transformations employed cannot be arbitrary. However this method works for the normal choice of the isometric transformations (four reflections and four rotations). The condition is that *the isometric transformations should not 'mix up' different range blocks that constitute a domain block.*

As this method converges in a *specific number* of iterations, this can be also be looked upon as a systematic procedure for *solving* the equation

$$\hat{\boldsymbol{A}} = w(\hat{\boldsymbol{A}}) \tag{2.16}$$

for the fixed point $\hat{\boldsymbol{A}}$.

Baharav et al [41] proposed a coder in which no isometric transformations are allowed. Domain blocks should always be twice the size of the range blocks and they should always overlap by the size of the range blocks. For such a coder, the iteration can be performed on an initial image where each pixel represents a range block, till convergence is achieved. The final image can then be obtained by a non-iterative expansion algorithm. However the limitation that no isometric transformations are permitted is a very serious one.

2.6 Wavelet-Fractal Coders

Wavelets and fractals have very intimate links. A strictly fractal object is similar at different scales, and the wavelet transform is a tool for analyzing objects at different scales. Thus under a proper choice of scales, the wavelet transform of a strictly fractal object will have a lot of redundancy. Bogdan [42] showed that the IFS representation of a function and the wavelet representation are very similar. A wavelet function $f(x)$ satisfies a two-scale equation

$$f(x) = \sum_k c_k f(2x - b_k) \tag{2.17}$$

while the IFS representation of a function $f(x)$ is of the form

$$f(x) = \sum_k c_k w(x - k) f(2x - b_k) \tag{2.18}$$

where $w(x - k)$ is a window function. (In the case of images it is the 'cut' operator which cuts out part of the image to form the domain block). Bogdan proposed a coder where (2.18) is solved for various levels of the pyramidal decomposition [7] of an image.

Krupnik [43] and Rinaldo [44] independently proposed a coder that worked on the subband decomposition (or wavelet decomposition) of the image [45]. In this method the image is decomposed, at every level, into four subbands, by column wise filtering followed row wise filtering by the low-pass and high-pass filters of a pair of quadrature mirror filters (QMF), followed by decimation. A pictorial representation of the decomposition into various subbands is given in Figure 2.4. The four subbands thus obtained, viz., LL,LH,HL and HH, are each one fourth the size of the original image. The LL subband is subjected to further decomposition to obtain LL1,LH1,HL1 and HH1. At the $r+1^{th}$ level of decomposition we get LLr,LHr,HLr and HHr. If n levels of decomposition are used, (typically for 512×512 images $n = 5$) then LL4 is coded directly. The other blocks of this level are predicted from LL4 and the prediction error is coded. The blocks of lower

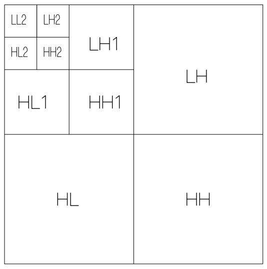

Figure 2.4: Subband decomposition of an image.

levels (larger blocks) are split into range blocks. The block with the same orientation at a higher level supplies domain blocks (of the same size as the range blocks) for coding the range blocks. Normally all possible domain blocks are tested. If no suitable domain block is found, then either the range block is subdivided into four or the range block is coded by other means.

Chapter 3

A Fast Method for Block Matching

3.1 Introduction

IFS image compression, as we have seen in the previous Chapters, involves finding the best domain block and its isometric transformation for each range block. In other words we desire to find the best *codebook* block for each range block. This method requires a least squares regression (LSR) for each comparison to determine the scale factor and the offset, followed by determining the error of the approximation, and then choosing the codebook block that yields the minimum error. In this Chapter we present a modified method, wherein the best fit codebook block can be determined in only one inner product of two vectors of length N (where N is the number of pixels in each range and codebook blocks) for each comparison.

Normally, the range blocks $\boldsymbol{R}_i$ and the domain blocks $\boldsymbol{D}_k$ are square matrices of pixel values, of size $r \times r$ and $nr \times nr$ respectively, where n is an integer greater than or equal to 2. Contraction of a domain block is achieved by partitioning it into r^2 sub-blocks of size $n \times n$ and assigning a single pixel value to each sub-block equal to the average of the pixel values within it. A set of t blocks is generated from each contracted domain block through isometric transformations. Normally 8 isometric transformations ($t = 8$) consisting of 4 reflections and 4 rotations are chosen [8]. Thus an enlarged pool of transformed blocks, called codebook blocks $\boldsymbol{C}_j, j = 1 \cdots N_d t$ is generated from the original pool of domain blocks $\boldsymbol{D}_k, k = 1 \cdots N_d$. Each range block $\boldsymbol{R}_i$ is then approximated by an affine transformation of a codebook block

$$\hat{\boldsymbol{R}}_i = s_i \boldsymbol{C}_j + o_i, i = 1, \cdots, N_r, \tag{3.1}$$

the choice of the matching codebook block $\boldsymbol{C}_j$ and the parameters s_i and o_i being made so as to minimize the approximation error $d(\boldsymbol{R}_i, \hat{\boldsymbol{R}}_i)$. Error minimization is usually achieved using LSR [16]. A simpler procedure based on the inner product of the range and codebook blocks is described below.

3.2 Block Matching by Inner Product

Let each range block be ordered as a vector by scanning. Let $\boldsymbol{r}_i$ be the zero mean range vector corresponding to the range block $\boldsymbol{R}_i$, obtained by subtracting the mean value of $\boldsymbol{R}_i$ from all the elements. Let $\boldsymbol{c}_j$ denote the zero-mean codebook vector obtained similarly from the codebook block $\boldsymbol{C}_j$. Both $\boldsymbol{r}_i$ and $\boldsymbol{c}_j$ are N-dimensional vectors, where $N = r^2$. The problem at hand is to choose, for a given i, the codebook vector $\boldsymbol{c}_j$ and the scalars s_i^j and o_i^j so as to minimize the error

$$\varepsilon_i^j = d(\hat{\boldsymbol{r}}_i, \boldsymbol{r}_i) = d(s_i^j \boldsymbol{c}_j + o_i^j, \boldsymbol{r}_i) = \langle \boldsymbol{r}_i - s_i^j \boldsymbol{c}_j - o_i^j, \boldsymbol{r}_i - s_i^j \boldsymbol{c}_j - o_i^j \rangle. \tag{3.2}$$

The symbol $\langle ., . \rangle$ denotes the scalar product of two vectors. Since both $\boldsymbol{r}_i$ and $\boldsymbol{c}_j$ are zero-mean vectors, it is obvious that

$$o_i^j = 0 \tag{3.3}$$

for minimizing ε_i^j. Putting $o_i^j = 0$ in (3.2) and equating the partial derivative of ε_i^j with respect to s_i^j to zero, we get

$$s_i^j = \frac{\gamma_{ij}}{\sigma_{c_j}^2} \tag{3.4}$$

where

$$\gamma_{ij} = \langle \boldsymbol{r}_i, \boldsymbol{c}_j \rangle, \sigma_{c_j} = \langle \boldsymbol{c}_j, \boldsymbol{c}_j \rangle \tag{3.5}$$

Substituting the value of s_i^j given by (3.4) into (3.2), we get

$$\varepsilon_i^j = \langle \boldsymbol{r}_i - \frac{\gamma_{ij}}{\sigma_{c_j}^2}\boldsymbol{c}_j, \boldsymbol{r}_i - \frac{\gamma_{ij}}{\sigma_{c_j}^2}\boldsymbol{c}_j \rangle = \sigma_{r_i}^2 - \frac{\gamma_{ij}^2}{\sigma_{c_j}^2}, \tag{3.6}$$

where ${\sigma_{r_i}}^2 = \langle \boldsymbol{r}_i, \boldsymbol{r}_i \rangle$. Equation (3.6) gives the minimum approximation error of $\boldsymbol{r}_i$ for a given $\boldsymbol{c}_j$. For minimizing ε_i^j over all $\boldsymbol{c}_j$, we have to find the codebook block for which $\frac{\gamma_{ij}^2}{\sigma_{c_j}^2}$ is maximum. If we normalize all codebook blocks by dividing them by σ_{c_j}, we just have to find the maximum of $|\langle \boldsymbol{r}_i, \tilde{\boldsymbol{c}}_j \rangle|$, where $\tilde{\boldsymbol{c}}_j = \frac{\boldsymbol{c}_j}{\sigma_{c_j}}$.

3.3 Limiting Scale Factor

It is normal practice to limit the scale factor s to some value s_{max}. Also when hierarchical coding is employed, we fix a tolerance, ε_{max}, for the approximation error. The two conditions can be mathematically stated as

$$\sigma_{r_i}^2 - \gamma_{ij} s < \varepsilon_{max}, \tag{3.7}$$

and

$$s < s_{max}. \tag{3.8}$$

From Schwartz's inequality, $\gamma_{ij}^2 \leq \sigma_{c_j}\sigma_{r_i}$. Therefore

$$\sigma_{r_i}^2 - \sigma_{c_j}\sigma_{r_i} s_{max} < \varepsilon_{max} \tag{3.9}$$

or

$$\sigma_{r_i} - \sigma_{c_j} s_{max} < \frac{\varepsilon_{max}}{\sigma_{r_i}}. \tag{3.10}$$

We therefore check for the validity of the above condition before we calculate γ_{ij} (which involves N multiplications). This reduces the number of domain blocks searched.

3.4 Quantization of Scale Factor

The scale factor s has to be quantized finally (normally to 5 to 7 bits) [46]. Therefore it would be better if the quantization error is taken into account before the choice of the best codebook block is made. Taking quantization error into account, the total error becomes

$$\varepsilon = \langle \boldsymbol{r} - (s + \triangle s)\boldsymbol{c} \ , \ \boldsymbol{r} - (s + \triangle s)\boldsymbol{c} \rangle. \tag{3.11}$$

After a little algebra,

$$\varepsilon = \sigma_{r_i}^2 - \frac{\gamma_{ij}^2}{\sigma_{c_j}^2} + \triangle s^2 \sigma_{c_j}^2. \tag{3.12}$$

Thus now we choose the codebook block for which $\frac{\gamma_{ij}^2}{\sigma_{c_j}^2} - \triangle s^2 \sigma_{c_j}^2$ is maximum.

Chapter 4

FFT Based Block Matching

4.1 The Method

Having seen in Chapter 3 that block matching involves just one inner product of the range and codebook vectors, we now present a method which reduces the computational complexity to less than N multiplications per comparison on an average. This is done by choosing the isometric transformations of the shrunken domain blocks to be different from the usual eight (4 rotations and 4 reflections) [8]. We convert each shrunken domain block into a vector using a suitable scanning procedure. We choose a set of $N = r^2$ transformations of each shrunken domain block by inverse scanning all possible circular shifts of the corresponding vector. Now the inner product of a range block with each member of a set of N codebook blocks derived from the same domain block is obtained by circular correlation of the range vector and the parent shrunken domain vector. This would involve N^2 multiplications. However we can implement the circular correlation using the FFT algorithm to reduce the number of multiplications to the order of $N \log_2 N$. Thus, the average number of multiplications per comparison is reduced from N to approximately $\log_2 N$.

Note that the number of isometric transformations used in this method is much more than the usual 8. For example, for 8×8 range blocks, we permit 64 transformations. We can also use the flipped or reverse scan vector whose Fourier transform is just the complex conjugate of the Fourier transform of the original scan vector to yield 64 more transformations. In general for range block size $r \times r$, we permit r^2 or $2r^2$ transformations of each shrunken domain block. As the number of transformations of each domain block is increased we can hope to reduce the number of domain blocks (to keep the number of codebook blocks constant) *provided all transformations are 'useful'.*

If $\boldsymbol{r} \in \Re^N$ is a range vector and $\boldsymbol{d} \in \Re^N$ is a shrunken domain vector (assume that the means are subtracted from $\boldsymbol{r}$ and $\boldsymbol{d}$ and further that $\boldsymbol{d}$ is normalized to have unit norm), the inner products of $\boldsymbol{r}$ with all the circular shift transformations of $\boldsymbol{d}$ can be written as a vector $\boldsymbol{p} \in \Re^N$:

$$\boldsymbol{p} = \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{d}).\mathcal{F}^*(\boldsymbol{r})), \tag{4.1}$$

where $\mathcal{F}$ stands for the DFT and $\mathcal{F}^{-1}$ for the IDFT, and the vector $\mathcal{F}(\boldsymbol{d}).\mathcal{F}^*(\boldsymbol{r})$ is obtained by multiplying the corresponding elements of the vectors $\mathcal{F}(\boldsymbol{d})$ and $\mathcal{F}^*(\boldsymbol{r})$. The inner products of $\boldsymbol{r}$ with all the circular shift transformations of the flipped vector $\boldsymbol{d}_r$ obtained from $\boldsymbol{d}$ can be represented as,

$$\boldsymbol{p}_{\mathrm{r}} = \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{d}).\mathcal{F}(\boldsymbol{r})). \tag{4.2}$$

Let $\tilde{\boldsymbol{p}}$ denote a vector of $2N$ elements, obtained by appending the elements of $\boldsymbol{p}_r$ to those of $\boldsymbol{p}$. The best suited transformation is given by the index of the element of $\tilde{\boldsymbol{p}}$ with the largest magnitude.

4.1.1 Implementation

The above scheme of FFT-based block matching was tried on a variety of 512×512 images. Among the N! possible scan orders, many were tried out, but only two of them were found to be useful, in terms of the SNR of the decoded image. The first one is the normal scanning method of stacking the rows of a block to form a single row vector. The second method is the Hilbert scan ordering of pixels of a block [47]. The Hilbert scanning of a 4×4 block is shown in Figure 4.1.

The 512×512 image was divided into 4096 range blocks of size 8×8 and 256 non-overlapping domain blocks of size 32×32. As explained in the previous section, 128 transformations of each shrunken domain block were permitted, making the number of codebook blocks equal to 32768. Scaling down of the domain blocks to the size of the range blocks was done by averaging sixteen pixels to one. Comparison of the range and codebook blocks was done by computing $\boldsymbol{p}$ and $\boldsymbol{p}_r$ using (4.1) and (4.2). The IFS code

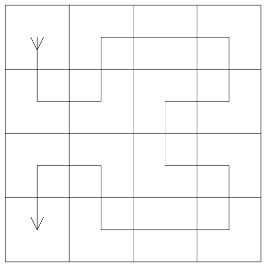
Figure 4.1: Hilbert curve scanning of a 4×4 block.

for each range block consists of:

- the scale factor (5 bits),
- mean of the range block (7 bits),
- address of one of the 256 domain blocks (8 bits), and
- transformation used (7 bits).

The results obtained by this method are compared with those obtained by the conventional (Jacquin's) technique using 3969 domain blocks of size 16×16 (adjacent domain blocks overlapping by the size of the range block), using the normal 8 transformations (four rotations and four reflections). For a fair comparison, we have used approximately the same number of codebook blocks for both the methods. which will lead to the same compression ratio. (The number of codebook blocks for Jacquin's method is $3969 \times 8 = 31752$). The results of the three methods, viz., FFT-based block matching (normal scan), FFT-based block matching (Hilbert scan), and Jacquin's method, with exhaustive search of the domain blocks, are presented in Table 4.1. It is seen that the SNR's obtained for FFT-based comparison methods are slightly lower than those of the conventional method, indicating that not all the circular shift transformations are 'useful' for representing image blocks. Also, among the two methods employing FFT-based block matching, the method using Hilbert scan ordering of pixels yields marginally better results. However, while the conventional method requires 31752×64 multiplications per range block, the number of multiplications required by the FFT-based methods is about 25 times less, as we will see in the next section.

The decoded 'Lena' image using the above procedure for Hilbert scan ordering of pixels in the domain blocks is shown in Figure 4.2 (a). The SNR is 29.12 dB. Figure 4.2 (b) shows the decoded image at an SNR of 28.58 dB for the normal scan. Figure 4.2 (c) shows the decoded 'Lena' image at an SNR of 30.36 dB, using Jacquin's Method.

4.2 Complexity of FFT-Based Block Matching

Among the various FFT algorithms, the split-radix FFT algorithm [48, 49, 50, 51] is known to have the least computational complexity. The number of multiplications required by this algorithm is close to the theoretical minimum [50, 51] and there is little room for further reduction of its computational complexity. For a real vector in $\Re^N$ (where N is a power of 2) the number of multiplications required by the split-radix algorithm is given by [49]

$$\mu_r(N) = \frac{N}{2}(\log_2 N - 3) + 2. \tag{4.3}$$

Specifically, for $M = 64$ the number of multiplications required is 98. Correlation of one range vector with one shrunken domain vector and its time reversed version, therefore, involves (see (4.1), (4.2))

1. Multiplication of the DFT's of the range and domain vectors and multiplication of the DFT of the range vector with the conjugate of the DFT of the domain vector. For real vectors of length $N = 64$, this can be implemented using $32 \times 4 = 128$ multiplications.

2. Two IFFTs. This would involve $98 \times 2 = 196$ multiplications.

Thus, each range vector is compared with 128 codebook vectors with just $128 + 196 = 324$ multiplications. On the other hand comparison of each range vector with 128 codebook vectors by inner product would require $64 \times 128 = 8192$ multiplications. Thus the FFT based methods provide a 25 fold reduction in the complexity of the block matching process. For a general N (which is a power of 2) the complexity reduction factor is given by

$$c_r = \frac{2N^2}{2[\frac{N}{2}(\log_2 N - 3) + 2] + 2N}. \tag{4.4}$$

Image	JM	FFT-NS	FFT-HS
Lena	30.36	28.58	29.12
Baboon	24.87	24.65	25.13
Boats	30.05	28.25	29.13
Peppers	31.85	28.56	29.45

Table 4.1: Comparison of SNR of decoded images for Jacquin's method (JM), FFT-based block matching - normal scan (FFT-NS) and Hilbert scan (FFT-HS).

For $N = 256$, $c_r = 70$; for $N = 16$, $c_r = 10$.

4.3 Faster Suboptimal Alternative

The block matching process can be further speeded up if we first check for a match in the magnitude DFT of the range and domain blocks. It should be noted that, all the codebook vectors obtained by circular shifting of a domain vector have the same magnitude DFT. (In addition their phase differences are linear). We can therefore make an initial comparison of the DFT magnitude to choose only those domain vectors that have a magnitude DFT close to that of the range vector. The scalar product of the two vectors is a measure of the degree of their closeness. Thus, in the case of 8×8 range blocks, we have to compute the scalar product of the 32 element DFT magnitude vector corresponding to each range block (due to the symmetry property of the DFT of a real vector, only half the magnitude coefficients are independent) with 256 vectors (each of length 32) corresponding to the 256 domain blocks. Correlation of a range vector can be carried out with only a small number $(5 - 10)$ of domain vectors whose magnitude DFT's are the closest to that of the range vector. Though a good match in the magnitude DFT, does not imply a good overall match, simulations show that even if only five to ten best matches are retained from the 256 domain blocks, there is not much difference in the quality of the match obtained in comparison with the exhaustive search method involving correlation with *all* the domain blocks. For 8×8 range blocks, computation of the scalar product of two magnitude DFT vectors involves 32 multiplications. Hence, if k best matches in the magnitude DFT are retained for correlation, the total number of multiplications per range block is $32 \times 256 + 324k$. The corresponding figure is 324×256 if correlation is done with all 256 domain blocks. Thus if If 5 best matches in magnitude Fourier domain are retained $(k = 5)$, the computational complexity is reduced by about 8 times. Overall, this method

Figure 4.2: The decoded 'Lena' image with (a) (top left) FFT-based block matching - Hilbert scan, (b) (top right) normal scan, and (c) (bottom) Jacquin's method.

k	Lena (NS)	Lena (HS)	Peppers (NS)	Peppers (HS)
1	26.12	28.23	26.05	28.21
2	26.55	28.34	26.45	28.52
3	27.11	28.56	26.65	28.72
5	27.32	28.92	27.01	29.21
10	28.04	29.01	27.52	29.35
20	28.46	29.09	28.10	29.43
256	28.58	29.12	28.25	29.45

Table 4.2: SNR of reconstructed images for various choices of the number k of best matching domain blocks retained. NS:normal scan, HS: Hilbert scan.

is therefore faster by $8 \times 25 = 200$ times compared to Jacquin's method. The results for various values of k are tabulated for both normal scan and Hilbert scan in Table 4.2. It can be seen that for the case of Hilbert scan, even choosing only the best matching domain block ($k = 1$) does not drastically affect the PSNR of the image. This is probably due to the fact that due to the nature of ordering of pixels in the Hilbert scan, the phase dependence is reduced (ie, the difference between vectors of adjacent shifts is not as high as in the case of the normal scan). Thus with the choice of the Hilbert scan for ordering of pixels, Fourier domain block matching, followed by FFT based correlation of chosen domain and range blocks, yields reasonably good quality decoded images with a drastic reduction in encoding complexity.

4.4 Application of FFT-Based Block Matching for Encoding Errors

As seen in the previous section, though FFT-based block matching can significantly speed up the encoding process of fractal image compression, the transformations used are not always 'useful'. Therefore to get the same quality of the encoded image as the method using the regular transformations, we will have to sacrifice compression ratio (by using more number of domain blocks and hence increasing the number of bits required for coding the domain block addresses). However we see that the circular shift transformations used in FFT-based block matching are useful for coding errors. This is due to the fact that errors usually occur as sharp features near the edges of the image, and these sharp features may occur anywhere in a block. Hence if all circular shifts of a block are the permitted transformations, all the transformations may be equally useful.

In this section we present a hybrid scheme of image compression in which the first level of coding is the regular fractal compression scheme (using 8 transformations) applied to the decimated image. The image is then decoded to full size. We then code the errors by FFT-based block matching. The implementation details are as follows. An image of size 512×512 is decimated by averaging to the size of 128×128, which is coded using range blocks of size 4×4 and domain blocks of size 8×8, overlapping by the size of the range block. The IFS code is decoded to obtain a 512×512 image. The 32×32 block of mean values of the range blocks (which is transmitted as part of the IFS code) is the source of domain blocks for coding errors. From the 32×32 block (which is actually the image decimated by a factor of 16) we form a pool of 49 domain blocks (All 8×8 blocks with an overlap of 4 pixels in both directions). The mean is subtracted from each domain block and it is normalized, scanned (normal scanning) and Fourier transformed. The 512×512 error image is divided into 8×8 blocks. If the mean of an error block is over a predecided threshold, we code its mean. If the variance of the error block is over a predecided threshold, the block is scanned, Fourier transformed and correlated with all the domain blocks to get the best match. If the error after the matching is still over the threshold, the residual error is again Fourier transformed and correlated with the domain blocks.

The details of the performance of the compression scheme on the Lena image is as follows. The first stage of compression yielded the image of Figure 4.3 (a) at an SNR of 24.7 dB. The IFS code, after entropy coding required about 2760 bytes (compression ratio of 95). For each of the 4096 error blocks of size 8×8 the following codes were used

- 1 bit to describe if the mean of the error block is to be coded. This was coded using 400 bytes.

- 3 bits to code the means of the blocks indicated by item 1. Note that this is not required for all the error blocks. This, after entropy coding required about 700 bytes.

- 2 bits to indicate the number of levels of residual error coding. 0-no error coding, 1- 1 level of error coding, 2-2 levels of error coding, and 3- 3 levels of error coding. For the 'Lena' image 896 blocks had three levels of coding, 525 blocks required two levels and 611 blocks required one level of coding. This was entropy coded to 890 bytes.

k	Lena	Boats
1	30.52	29.26
2	30.78	29.52
3	30.92	29.78
5	31.14	30.02
10	31.23	30.10
49	31.25	30.15

Table 4.3: SNR of reconstructed images of the multistage compression scheme for various choices of the number k of best matching domain blocks retained.

- 5 bits for the scale factors of each level. After entropy coding, this required about 2500 bytes.
- 6 bits for the domain block address required a total of about 2800 bytes after entropy coding.
- 7 bits for the transformation, 3500 bytes.

The overall requirement was thus 13550 bytes, or a bit rate of 0.41 bpp at an SNR of 31.25 dB. The decoded image is shown in Figure 4.3 (b). As in the previous section, performing the comparison of the error range blocks and the domain blocks in the magnitude Fourier domain with a view of restricting the number of domain blocks searched, has been found to significantly reduce encoding time with only slight detoriation in the quality of the decoded image. The PSNR for two images compressed under this scheme for various choices of the number of domain blocks retained (k) for correlation with the error (range) block is shown in Table 4.3

From Table 4.3 it is seen that even choosing the best match from the magnitude Fourier domain does not drastically reduce the SNR. If the five best matches are chosen, the difference is too small to be noticed.

Figure 4.3: (a) (Top) ‘Lena’ image at compression ratio of 95, and (b) (bottom) after error coding (compression ratio 19.5, SNR 31.25 dB).

Chapter 5

A Class of Feature Highlighting Transforms

5.1 Introduction

Edges carry a significant amount of information in an image, and preservation of edges is a primary requirement of any image coding scheme. Since an edge is not necessarily associated with high pixel value, the PSNR of an image coding scheme does not give a true indication of the latter's ability to preserve edges.

The quality of a fractal coding scheme may be improved significantly if coding is preceded by a pre-processing scheme for edge enhancement. Towards this end we define in this Chapter, a class of transformations called Feature Highlighting Transforms (FHT). This transform redistributes the energy in a signal in such a fashion that the occurrence of any specified features in the signal domain is highlighted by a peak at the corresponding location in the transform domain. In the context of images, the feature of interest is an edge, and the edges are highlighted by large values of the corresponding transform coefficients. If the FHT transformed image is coded, the chances of losing an edge are lower than when the coding is done in the image domain. For reconstructing the image, decoding is followed by an inverse FHT. An important feature of the FHT is that the direct an inverse transforms can be evaluated using FFT. Hence, the additional computational load involved in coding via FHT is small.

5.2 Feature Highlighting Transforms

Let $\boldsymbol{b}_0, \boldsymbol{b}_1, \cdots, \boldsymbol{b}_{N-1}$ be a set of mutually orthogonal unit-norm vectors spanning $\Re^N$, so that

$$\boldsymbol{b}_i^T \boldsymbol{b}_j = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Let $\boldsymbol{x} = [x(0)\ x(1)\ \cdots\ x(N-1)]^T$ be an arbitrary element of $\Re^N$. The unitary transformation defined by

$$\tilde{\boldsymbol{x}} = [\boldsymbol{b}_0\ \boldsymbol{b}_1\ \cdots\ \boldsymbol{b}_{N-1}]^T \boldsymbol{x} \tag{5.2}$$

conserves energy, i.e.

$$\tilde{\boldsymbol{x}}^T \tilde{\boldsymbol{x}} = \boldsymbol{x}^T \boldsymbol{x}\ \forall\ \boldsymbol{x} \in \Re^N. \tag{5.3}$$

It follows from (5.2) that the elements of $\tilde{\boldsymbol{x}}$ can be expressed as the inner products of $\boldsymbol{x}$ with the basis vectors $\boldsymbol{b}_n$, i.e.

$$\tilde{x}(n) = \boldsymbol{x}^T \boldsymbol{b}_n,\ n = 0, 1, \cdots, N-1. \tag{5.4}$$

Since the matrix $[\boldsymbol{b}_0\ \boldsymbol{b}_1\ \cdots\ \boldsymbol{b}_{N-1}]$ is unitary it also follows that the inverse transformation is given by

$$\begin{aligned} \boldsymbol{x} &= [\boldsymbol{b}_0\ \boldsymbol{b}_1\ \cdots\ \boldsymbol{b}_{N-1}]\tilde{\boldsymbol{x}} \\ &= \sum_{n=0}^{N-1} \tilde{x}(n) \boldsymbol{b}_n \end{aligned} \tag{5.5}$$

Let the signal vector $\boldsymbol{x}$ contain a feature of interest at the n^{th} location. We seek a unitary transformation of $\boldsymbol{x}$ such that the n^{th} transform coefficient $\tilde{x}(n)$ is significantly large. More generally, if the feature of interest occurs at more that one location in $\boldsymbol{x}$, all the corresponding transform coefficients should be large compared to to the rest. Such a transformation will be called a Feature Highlighting Transform (FHT).

Consider, for example the time series $\boldsymbol{x} \in \Re^N$ shown in Figure 5.1(i). Let the 'feature of interest' in the time series be the occurrence of two negative going pulses sandwiched between two positive ones. (This feature occurs at four places in $\boldsymbol{x}$.) The feature vector is defined as a unit-norm vector $\boldsymbol{f} \in \Re^N$, containing a replica of the desired feature in the beginning and zeroes elsewhere, as shown in Figure 5.1(ii). Consider the set of N vectors $\boldsymbol{f}_0, \boldsymbol{f}_1, \cdots, \boldsymbol{f}_{N-1}$, where $\boldsymbol{f}_n$ is the vector obtained by circularly shifting the elements of $\boldsymbol{f}_0$ through n places to the right. Let $y(n)$ be the inner product of $\boldsymbol{x}$ and $\boldsymbol{f}_n$

$$y(n) = \boldsymbol{x}^T \boldsymbol{f}_n, \; n = 0, 1, \cdots, N-1. \tag{5.6}$$

This inner product has a large value if and only if the feature of interest is present at the n^{th} location of $\boldsymbol{x}$. In other words the transform

$$\boldsymbol{y} = [y(0) \; y(1) \; \cdots \; y(N-1)] = [\boldsymbol{f}_0 \; \boldsymbol{f}_1 \; \cdots \; \boldsymbol{f}_{N-1}]^T \boldsymbol{x} \tag{5.7}$$

has peaks at the locations of the feature of interest in the signal $\boldsymbol{x}$. However the transformation defined by (5.7) is not easily invertible since its basis vectors $\boldsymbol{f}_n$ are not mutually orthogonal. This drawback can be overcome if we can find a vector $\boldsymbol{b}$ such that

1. it is close to the feature vector $\boldsymbol{f}$, and
2. all its circular shifts are mutually orthogonal.

We now show that all circular shifts of the impulse response of any all-pass filter [52] are mutually orthogonal. Let $\boldsymbol{h} \in \Re^N$ be the impulse response of an all-pass filter and let $\boldsymbol{H} = \mathcal{F}(\boldsymbol{h})$ be the corresponding transfer function. It follows that $| H(n) | = 1$ for $n = 0, 1, \cdots, N-1$, and hence

$$(\boldsymbol{H}.\boldsymbol{H}^*) = [1, 1, \cdots, 1]^T, \tag{5.8}$$

where $\boldsymbol{H}.\boldsymbol{H}^*$ denotes the vector obtained by multiplying the corresponding elements (Hadamard product) of $\boldsymbol{H}$ and $\boldsymbol{H}^*$. Taking the IDFT of both sides of (5.8) we get

$$\mathcal{F}^{-1}(\boldsymbol{H}.\boldsymbol{H}^*) = [1, 0, 0, \cdots, 0]^T \tag{5.9}$$

but $\mathcal{F}^{-1}(\boldsymbol{H}.\boldsymbol{H}^*)$ is nothing but the circular autocorrelation of the impulse response vector

$\boldsymbol{h}$. It follows that all circular shifts of $\boldsymbol{h}$ are mutually orthogonal [53, 54].

As the phases ϕ_n $(n = 0, 1, \cdots, N-1)$ of the elements of $\boldsymbol{H}$ can be arbitrary, we have infinitely many choices for the vector $\boldsymbol{h}$ with mutually orthogonal circular shifts. The trivial choice of $\phi_n = 0 \; \forall \; n$ leads to $\boldsymbol{h} = \boldsymbol{e}_0 = [1 \; 0 \; \cdots \; 0]^T$. The vector $\boldsymbol{e}_0$ along with its circular shifts form the standard basis for $\Re^N$. The transformation corresponding to this set of basis vectors is the identity transformation. Other choices of the phases of ϕ_n yield other sets of orthonormal basis vectors corresponding to nontrivial transformations. For a feature highlighting transform, the first basis vector is chosen to be the all-pass impulse response that is closest to the appropriate feature vector.

Popat et al [55] have used dispersive FIR all-pass filters to make any memoryless source appear Gaussian, thereby facilitating efficient Lloyd-Max quantization. Strube [56] has used an all-pass filter in an ADPCM system to disperse pitch pulses over time to reduce quantizer overload distortion. He has also suggested a technique for constructing an all-pass filter with a desired impulse response. However for the case of all-pass impulse response of finite duration, in the next Section, we outline a very simple procedure for constructing the closest basis vector (an all-pass impulse response) to any arbitrary desired feature vector.

5.3 Construction of Basis Vectors for FHT

We have seen in the previous Section that any all-pass impulse response vector of size N and all its circular shifts form an orthonormal basis for $\Re^N$. For highlighting a feature represented by the feature vector $\boldsymbol{f} = [f(0) \; f(1) \; \cdots \; f(N-1)]^T$ we need to find the all-pass impulse response that is closest to $\boldsymbol{f}$. In other words we need to minimize the error ε defined as

$$\varepsilon = \sum_{n=0}^{N-1} \mid h(n) - f(n) \mid^2, \tag{5.10}$$

where $h(n)$, $n = 0, 1, \cdots, N-1$ are the elements of the impulse response vector $\boldsymbol{h}$ of an all-pass filter. Since the transfer function of an all-pass filter can be written as

$$\boldsymbol{H} = [e^{j\phi_0} \; e^{j\phi_1} \; \cdots \; e^{j\phi_{N-1}}]^T, \tag{5.11}$$

it follows that

$$h(n) = \sum_{k=0}^{N-1} e^{j(\frac{2\pi kn}{N}+\phi_k)},\ n = 0, 1, \cdots, N-1 \tag{5.12}$$

Let

$$f(n) = \sum_{k=0}^{N-1} a_k e^{j(\frac{2\pi kn}{N}+\theta_k)},\ n = 0, 1, \cdots, N-1 \tag{5.13}$$

Hence the error ε is given by

$$\begin{aligned}
\varepsilon &= \sum_{n=0}^{N-1}\sum_{k=0}^{l-1}\sum_{k=0}^{N-1}[e^{j(\frac{2\pi kn}{N}+\phi_k)} - a_k e^{j(\frac{2\pi kn}{N}+\theta_k)}] \times [e^{-j(\frac{2\pi ln}{N}+\phi_l)} - a_l e^{-j(\frac{2\pi ln}{N}+\theta_l)}] \quad (5.14)\\
&= \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}(\sum_{n=0}^{N-1}[e^{j(\frac{2\pi(k-l)n}{N})}e^{j(\phi_k-\phi_l)} - a_l e^{j(\frac{2\pi(k-l)n}{N})}e^{j(\phi_k-\theta_l)}\\
&- a_k e^{j(\frac{2\pi(k-l)n}{N})}e^{j(\theta_k-\phi_l)} + a_k a_l e^{j(\frac{2\pi(k-l)n}{N})}e^{j(\theta_k-\theta_l)}])
\end{aligned}$$

Using the identity

$$\sum_{n=0}^{N-1} e^{j(\frac{2\pi(k-l)n}{N})} = \begin{cases} N & \text{for } k = l \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

for $k, l = 0, \cdots, N-1$, (5.14) is reduced to

$$\varepsilon = N[N - 2\sum_{k=0}^{N-1} a_k cos(\phi_k - \theta_k) + \sum_{k=0}^{N-1} a_k^2] \tag{5.16}$$

Hence, the error is minimized if we choose

$$\phi_k = \theta_k,\ k = 0, 1, \cdots, N-1 \tag{5.17}$$

In other words, the phase spectrum of $\boldsymbol{h}$ should be identical to that of the feature vector $\boldsymbol{f}$.

The following simple algorithm can be used for designing the basis vectors of the FHT.

1. Define the vector $\boldsymbol{f}$, the desired feature vector.

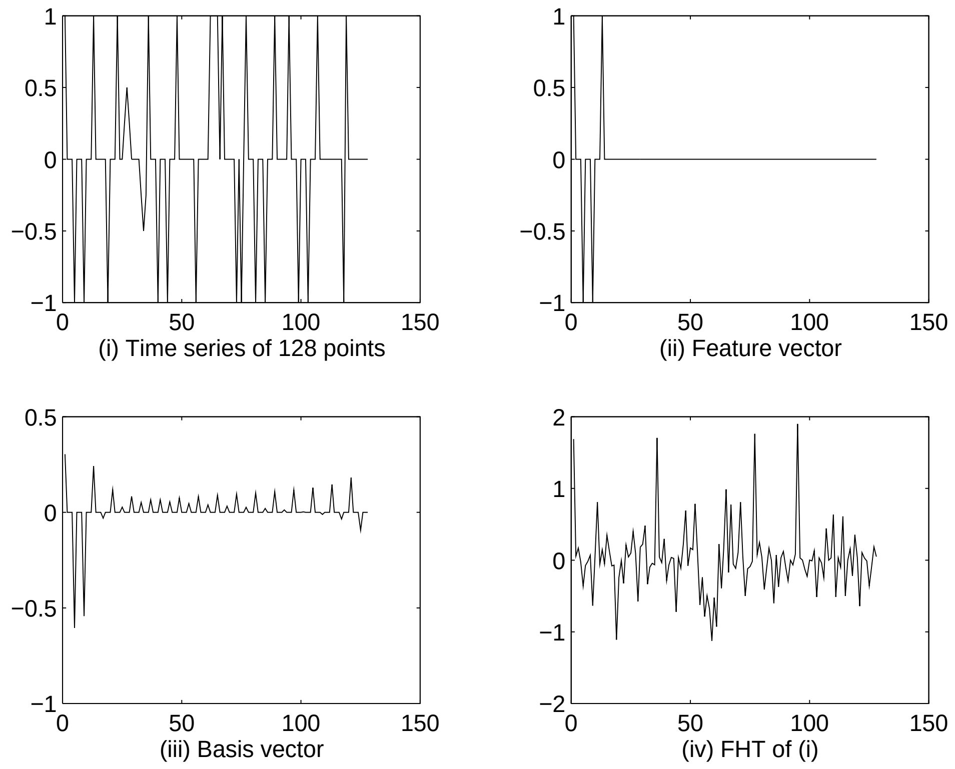

Figure 5.1: FHT for highlighting the feature of figure (ii). The ‘desired feature’, shown in (ii), occurs at four instances in (i) corresponding to the four leading transform coefficients in (iv); (iii) is the closest all-pass impulse response to the feature vector of (ii) and hence the basis vector used for the transformation.

2. Compute the DFT $\boldsymbol{F}$ of $\boldsymbol{f}$.

3. Retain the phase values $\phi_0, \cdots, \phi_{N-1}$ of $\boldsymbol{F}$, and reset all the amplitude coefficients to 1 to obtain the vector $\boldsymbol{H}$.

4. Compute the IDFT of $\boldsymbol{H}$ to obtain the vector $\boldsymbol{h}$.

5. Obtain the first basis vector $\boldsymbol{b}_0$ by setting $\boldsymbol{b}_0 = \boldsymbol{h}$. Obtain the remaining basis vectors $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{N-1}$ by circular shifting of $\boldsymbol{b}_0$ by $1, \cdots, N-1$ steps.

Figure 5.1(iii) shows the all pass impulse response vector that is closest to the feature vector in Figure 5.1(ii). The FHT of the signal in Figure 5.1(i), corresponding to this basis vector, is shown in Figure 5.1(iv).

5.4 Implementation of FHT and IFHT

The FHT and inverse FHT (IFHT) can be efficiently implemented using FFT by taking advantage of the fact that all the basis vectors are circularly shifted versions of the first basis vector. Specifically, we have

$$b_n(k) = b_0(k-n) = h(k-n);\ k,n = 0,\cdots,N-1. \tag{5.18}$$

Combining (5.4) and (5.18), we get

$$\tilde{x}(n) = \sum_{k=0}^{N-1} x(k)h(k-n) \tag{5.19}$$

and hence

$$\tilde{\boldsymbol{x}} = \mathcal{F}^{-1}(\mathcal{F}(x).\boldsymbol{H}^*). \tag{5.20}$$

From (5.6) and (5.18), we get

$$\begin{aligned} x(k) &= \sum_{n=0}^{N-1} \tilde{x}(n)b_n(k) \qquad (5.21)\\ &= \sum_{n=0}^{N-1} \tilde{x}(n)h(k-n), \end{aligned}$$

and hence

$$\boldsymbol{x} = \mathcal{F}^{-1}(\mathcal{F}(\tilde{\boldsymbol{x}}).\boldsymbol{H}). \tag{5.22}$$

(5.20) and (5.22) indicate that two N-point FFTs have to be performed to compute the FHT and an equal number to compute the IFHT.

5.5 Extension to Higher Dimensions

Extension of FHT to higher dimensions is conceptually straightforward. The extension can be readily demonstrated by considering a 2-dimensional signal or image of

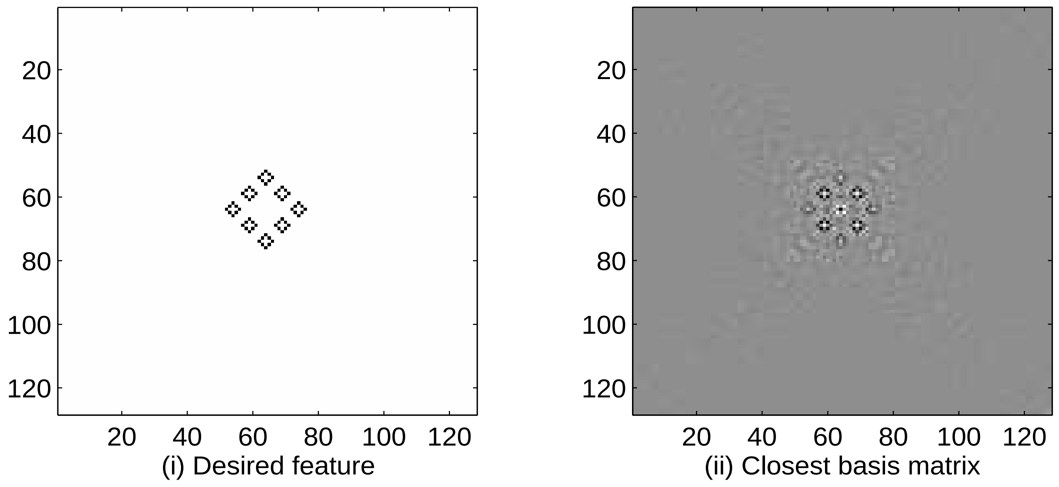

Figure 5.2: An example of 2-dimensional feature and basis vectors.

size $N \times N$. The image space $\Re^{N^2}$ is spanned by a set of N^2 basis matrices. We want the basis matrices to be mutually orthogonal and derivable from a single matrix through combinations of circular shifts in both directions. The first basis matrix is obtained by taking the 2-dimensional DFT of the feature matrix $\boldsymbol{F}$, resetting the magnitudes of all Fourier coefficients to unity while retaining the phases, and taking the 2-dimensional IDFT. Figure 5.2 gives an example of 2-dimensional feature matrix and the closest basis matrix. The other $N^2 - 1$ basis matrices are obtained by different shifts of the first basis matrix in both dimensions.

5.6 Use of FHT in Image Compression

The FHT used in conjunction with a fractal coding technique for low-bit-rate image compression. FHT is applied to the image before encoding in order to enhance the edges of of the image. Since the edges correspond to higher transform coefficients, they are better preserved in the coding-decoding process. The FHT transformed (edge-enhanced) image may be encoded using any coding scheme. After decoding the IFHT is performed to return to the original spatial domain. The feature matrix $\boldsymbol{F}$ chosen for edge enhancement has the same size as the image matrix, with $\boldsymbol{F}(0,0) = \boldsymbol{F}(1,1) = 1, \boldsymbol{F}(0,1) = \boldsymbol{F}(1,0) = -1$, and zero elsewhere. The corresponding basis matrices are obtained using the procedure described in the last section.

The original 512×512 'Lena' image, shown in Figure A.2 (a), was encoded and reconstructed using this technique. After transforming to the FHT domain, the encoding was done using the standard IFS compression scheme of Jacquin [8]. The FHT-domain image was segmented into 16×16 range blocks and 32×32 domain blocks. The domain blocks were shrunk to the size of the range blocks by averaging 4 pixels to form 1 pixel. The standard set of eight transformations (4 rotations and 4 reflections) was used to transform the domain blocks . 5 bits were used to store each scale factor, 6 bits for the mean of the range blocks, 10 bits for the domain block address and 3 bits for the transformation. resulting in a total of 24 bits per range block.

The fractal code was decoded to the original image size of 512×512, the inverse FHT was taken, and the resulting image smoothed along the edges by averaging three pixels. Figure 5.3 (a) shows the output image at a compression ratio of 95. Comparison of this with the image shown in Figure 5.3 (b), which is obtained the same way except that FHT was not used before the fractal coding (and thus no IFHT after the decoding), shows that the image of Figure 5.3 (a) has a much better reproduction of significant edges than that of Figure 5.3 (b). The improvement is especially noticeable near the tip of the hat and the nose of Lena. Figure 5.4 shows the blown up portions of the face (left) and the top edge of the cap (right). The images, in this figure, from top to bottom, are blown up portions of images of Figure 5.3 (a) (coding with edge-enhancement),5.3 (b) (coding without edge-enhancement) and Figure A.2 (a) (original 'Lena'). Images in Figure 5.3 have both been subjected to smoothing along the edges of the range blocks. While the blockiness in the top image is almost removed, it is still very much apparent in the bottom image. Overall the image of Figure 5.3 (a) is visually more acceptable than that in Figure 5.3 (b), though the SNR of the former (25.15 dB) is slightly less than that of the latter (25.72dB).

5.7 A Variant of the FHT

A disadvantage of using an all pass impulse responses as basis vectors is the non-smoothness of the basis vectors. Truncation of the transform coefficients may therefore cause annoying effects. Note that by using all pass functions, the FHTs sacrifice frequency resolution to achieve maximum spatial / temporal resolution (as the basis vectors are orthogonal to all shifts). Frequency transforms like DFT and DCT have no spatial

Figure 5.3: (a) (Top) decompressed ‘Lena’ image at compression ratio of 95 with FHT, (b) (bottom) without FHT.

Figure 5.4: Blowup of face and tip-of-hat of ‘Lena’. Top row- with FHT. Middle row- without FHT. Bottom row- original ‘Lena’.

/ temporal resolution, but have maximum possible frequency resolution. Wavelet transforms [4, 57] strike a compromise between these two extremes. Taking a cue from wavelet transforms we may sacrifice some spatial / temporal resolution (by relaxing the condition of orthogonality to all shifts) and get a better approximation of the 'desired' basis function (or smoother basis functions) . For the case of orthogonality to *alternate* shifts the magnitude each element of the DFT of the basis vector need not be unity. In this case $\mid B(k) \mid$, $k = 0 \cdots N-1$ just have to satisfy the condition [57]

$$|B(k)|^2 + |B(k + \frac{N}{2})|^2 = 2 \tag{5.23}$$

This gives us a little more freedom in choosing the magnitude of the DFT coefficients of the basis vectors. Using these basis vectors, $\frac{N}{2}$ coefficients of the transformation are generated as the even correlation coefficients of the data vector and $\boldsymbol{b}$. The other coefficients are obtained from the even correlation coefficients of the data vector with $\boldsymbol{g}$, where $\boldsymbol{g}$ is obtained by time reversing $\boldsymbol{b}$ (flipping the vector) and changing the sign of all odd coefficients [5]. Here $\boldsymbol{B}$ and $\boldsymbol{G}$ (where $\boldsymbol{b} \overset{\mathcal{F}}{\leftrightarrow} \boldsymbol{B}$ and $\boldsymbol{g} \overset{\mathcal{F}}{\leftrightarrow} \boldsymbol{G}$) are quadrature mirror filters (QMF) [53, 54]. Alternately $\boldsymbol{b}$ corresponds to the scaling function and $\boldsymbol{g}$ corresponds to the wavelet. Figure 5.5 shows an example where for a given feature vector, the closest basis vector (CBV) that is orthogonal to all shifts is a very poor approximation. However on relaxing the condition of orthogonality to every shift to orthogonality to alternate shifts we get the CBV that is a much better approximation of the feature vector.

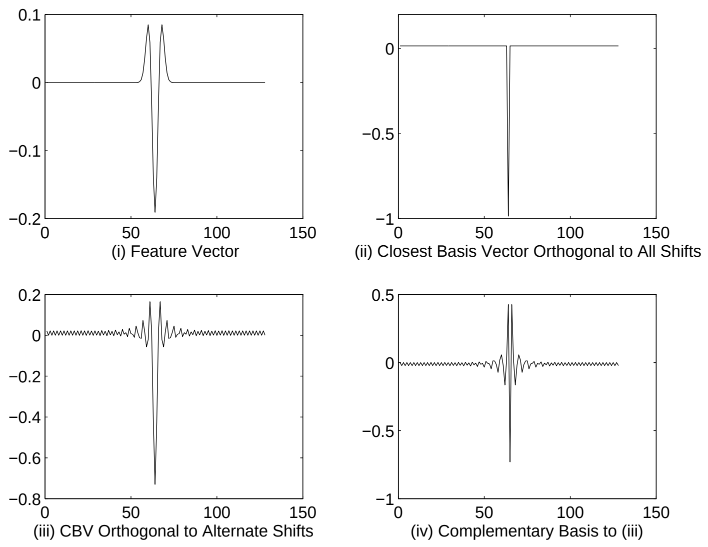

Figure 5.5: An example demonstrating tradeoffs between temporal resolution and the closeness of the basis vector to the feature vector.

Chapter 6

Interpolation of Images

6.1 Fractal Interpolation

Fractal, or IFS coding of images, lends itself naturally to interpolation. We have seen in Chapter 2 that the IFS code for each range block consists of four parameters,

1. domain block address,
2. isometric transformation,
3. offset or range block mean, and
4. scale factor.

The IFS code does not specify the number of pixels in each range block. Therefore the image can be decoded to any size, with the only limitation that the range blocks of the decoded image have edges that are whole numbers. For example an $M \times N$ image coded with range blocks of size $r \times r$ can be decoded as an image of size $\frac{M}{r}m \times \frac{N}{r}m$ with range blocks of size $m \times m$. In this Chapter we discuss some features of fractal interpolation of images and compare it with interpolation using DCT and subband interpolation.

One major disadvantage of interpolation using IFS is that it does not guarantee distortionless coding of the original image however much we increase the bit rate of the coding scheme. At the coding stage of the IFS we try to reduce the collage error

$$\varepsilon_c = \sum_{n=1}^{N_r} d(\boldsymbol{R}_i, \hat{\boldsymbol{R}}_i) \tag{6.1}$$

in order to reduce the reconstruction error

$$\varepsilon = d(\boldsymbol{A}, \hat{\boldsymbol{A}}) \tag{6.2}$$

where, $\boldsymbol{R}_i$ and $\hat{\boldsymbol{R}}_i$ denote, respectively, the i^{th} range block and its approximation, $\boldsymbol{A}$ is the original image, and $\hat{\boldsymbol{A}}$ is the reconstructed image. Collage theorem guarantees the closeness of $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}$ if the collage error ε_c is small. Similar to (1.15), the collage error too can be expressed in terms of the peak signal to collage error ratio, (PSN$_c$R) as

$$\text{PSN}_c\text{R} = 10\log_{10}\frac{(2^b-1)^2 MN}{\varepsilon_c}, \tag{6.3}$$

where b is the number of bits used to represent each pixel of the original image of size $M \times N$. It is meaningless to talk about the reconstruction error in the case of IFS interpolated images, since $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}$ are of different sizes. However, it is intuitively obvious that we should first have an IFS code with the minimum possible collage error to obtain interpolated images of good quality. Also, it is known that we can increase the fidelity of the IFS, by reducing the size of the range blocks. We would therefore expect that, for good interpolation, we should code the original image with very small range blocks and then decode the IFS code for an image of larger size.

Figure 6.1 (a) shows an interpolated image of size 512×512, starting with an original image of size 128×128 and using the IFS code with 2×2 range blocks. Figure 6.1 (b) shows the interpolated image of the same size using the IFS code with 4×4 range blocks. Though the IFS code (with 2×2 range blocks) used for generating the image in Figure 6.1 (a) has a PSN$_c$R of 51 dB, which is higher than the PSN$_c$R of 32 dB for the IFS code (with 4×4 range blocks) used for generating the image in Figure 6.1 (b), the interpolated image in Figure 6.1 (a) is obviously inferior to that in Figure 6.1 (b). Apparently, reduction of the collage error is not a sufficient condition for improving the quality of the interpolated image. The reason may be that the 2×2 range blocks are *too small to catch any meaningful feature of the image.* We should therefore try to improve the fidelity of the IFS code, without reducing the range block size. This can be done by coding a range block $\boldsymbol{R}$, with multiple domain blocks:

$$\hat{\boldsymbol{R}} = m_r + \sum_{i=1}^{n} s_i \boldsymbol{C}_i, \tag{6.4}$$

Figure 6.1: (a) (Top left) interpolated ‘Elephants’ image (512×512). The original 128×128 image was coded using 2×2 range blocks.(b) (Top right) coded using 4×4 range blocks. (c) (Bottom left) 4×4 range blocks with n of (6.4) equal to 8. (d) (Bottom right) $n = 8$.

where m_r is the mean of the range block, $\boldsymbol{C}_i$ are zero mean, normalized codebook blocks and s_i are the scale factors. For $n = 1$, (6.4) reduces to (2.15). There is one more advantage of using a larger range block size. As already stated, an IFS coded image of size $M \times N$ can be decoded to a size $m\frac{M}{r} \times m\frac{M}{r}$, where m is any positive integer and $r \times r$ is the size of the range blocks. Hence, the larger the value of r, the larger is the number of possible sizes to which the image can be decoded.

6.1.1 Convergence of the IFS

The use of multiple domain blocks makes the study of convergence of the IFS even more intricate. However if the scheme proposed by Oien and Lepsoy [40], and outlined in Section 2.5, is used, it can be easily seen that convergence is reached in a fixed number of iterations, irrespective of the scale factor, *even if multiple domain blocks are used.* We may therefore use IFS coding with the following restrictions:

- Domain blocks having an integral square number of range blocks.
- Decimation of domain blocks by averaging
- Choice of the 8 standard transformations consisting of reflections and rotations.

As in [40], for any arbitrary choice of the initial image, after the first iteration, the means of all range blocks are fixed as the second term in the right-hand side of (6.4) is zero mean. If the size of the domain blocks is 4 times that of the range blocks, the means of each quadrant of each range block is fixed after the second iteration. After the third iteration the means of each quadrant of each quadrant of each range block are fixed, and so on, till the mean (or just the value) of each pixel is fixed, at which point, convergence is achieved.

6.1.2 Results

The IFS interpolated 512×512 images of 'Elephants' using 4×4 range blocks with n (of (6.4)) equal to 4 (PSN$_c$R=50.5 dB) and 8 (52 dB) are shown in Figure 6.1 (c) and (d) respectively. Figure 6.2 shows other examples of interpolation of the 128×128 images shown in Figure A.1 to 512×512 images. While the images in Figure 6.2 (a) (PSN$_c$R=49dB), (b) (46 dB) and (c) (49dB) use 4×4 range blocks with $n = 4$, the image in Figure 6.2 (d) (48 dB) is the result of interpolation using 8×8 are blocks with $n = 16$. Though the 'Girl' image in Figure 6.2 (d) is more acceptable than that in Figure 6.2 (c), neither of them is of high quality. Though fractal interpolation adds a lot of details to the interpolated image, it also adds a lot of artifacts which are unacceptable. Particularly unacceptable artifacts are the speckle noise on smooth regions and the secondary edges (like the one near the face of the 'Girl' in Figure 6.2 (d). Also note that as n is increased, (or if the IFS is coded using more number of bits) one does not perceive more than a slight improvement in the quality of the interpolated image. For example, there is not much difference between the images of Figure 6.1 which are coded with increasing bit rates.

6.1.3 Interpolation as a Route to Low Bit Rate Fractal Coding

As we have seen in the previous section, the quality of the interpolated image does not improve much as bit rate of the IFS code is increased. It is observed that there is not much use in coding the images at the original size at more than 35 dB PSNR. However fractal interpolation can be useful for low bit rate coding. As an example, Figure 5.3 (b) was coded using 16×16 range blocks to achieve a compression ratio of about 95 with an SNR of 25.72 dB. However the image of Figure 4.3 (a) was obtained by coding the decimated version of the image and then reconstructed by interpolation to the original size. The compression ratio is again about 95 and the SNR is 24.85 dB. Though the SNR of the latter is lower than that of the one coded without decimation and interpolation, it is advantageous to use the latter scheme as the computational complexity is drastically reduced since, in effect, an 128×128 image is coded instead of a 512×512 image.

6.1.4 Feature Highlighting Transforms for Interpolation

We have seen in Chapter 5, as to how feature highlighting transforms may be used to significantly increase the quality of the decompressed image. We now outline a scheme where FHTs can be used for interpolation of images. For example, for interpolating an 128×128 image to the size of 512×512 , the input image is FHT transformed (with basis vectors of size 128×128) and then coded. The decoding is done to obtain a 512×512 image which is inverse transformed using a kernel of size 512×512. The forward transform kernel is obtained from a feature vector $[1 \ -1 \ 0 \ 0 \ \cdots \ 0] \in \mathcal{R}^{128}$. The inverse transform kernel is obtained from the feature vector $[1 \ -1 \ 0 \ 0 \ \cdots \ 0] \in \mathcal{R}^{512}$ (zero padding the

Figure 6.2: (a) (Top left) 512×512 interpolated 'Squirrels' image for $n = 4$. (b) (Top right) 'Cheetah' (c) (bottom left) and (d) (bottom right) 'Girl'. (a), (b) and (c) are results of interpolation with range block size of 4×4 and $n = 4$. (d) is obtained from range block size of 8×8 with $n = 16$.

forward transform feature vector). The basis vectors are obtained from the feature vectors as the nearest all pass impulse response to the feature vector, as outlined in Chapter 5. The result of this interpolation for the 128×128 'Lena' image (which was obtained by decimating a 512×512 Lena image) is shown in Figure 6.3 (a). The SNR of the decoded image at 512×512 is 24.53 dB. Figure 6.3 (b) shows another example of interpolation using FHT for the 128×128 squirrel image to a size of 512×512.

6.2 Comparison with Other Interpolation Schemes

6.2.1 Interpolation Using DCT

For this form of interpolation the DCT coefficients of the original image (scaled appropriately - for example for interpolation from 128×128 to 512×512, each of the 128×128 DCT coefficients are multiplied by 16) are taken as the low frequency coefficients of the DCT of the interpolated image. These coefficients are zero-padded and the inverse DCT is taken to obtain the interpolated image. The interpolated 'Elephants' image thus obtained is shown in Figure 6.4 (a). Though the quality of the image is slightly better than that in Figure 6.4 (b), obtained by linear interpolation, the DCT interpolated image lacks sharpness, unlike the fractal interpolated image. But the advantage of DCT interpolation is that no annoying artifacts are added to the image.

6.2.2 Interpolation Using Subband Decomposition

In this method the image to be interpolated is assumed to be the product of the subband decomposition of the interpolated image. (To be more specific, the subband obtained by both column and row wise filtering by the low pass component of the QMF pair, which is commonly referred to as the LL band). The interpolated image is obtaining by *re*constructing the image through subband reconstruction, in which the subbands LH,HL and HH are all set to zero. Figure 6.5 shows the method of constructing the interpolated image from the original image. Figures 6.4 (c) and (d) show the interpolated 'Elephants' and 'Cheetah' images of size 512×512. For both images, the low pass filter used is the Daubachies's filter [4] of length 6. Comparison with other forms of interpolation shows that the subband interpolation is perhaps the best method. The sharpness of the images is better than that for DCT interpolation or linear interpolation, and unlike fractal

Figure 6.3: (a) (Top) interpolated 'Lena' image with FHT. (Figure 4.3-(a) shows the the interpolated 'Lena' image without FHT). (b) (Bottom) interpolated 'Squirrel' image with FHT.

Figure 6.4: (a) (Top left) DCT interpolation of 128×128 'Elephants' image to 512×512 image. (b) (Top right) linear interpolation. (c) (Bottom left) subband interpolation of 'Elephants' image. (d) (Bottom right) subband interpolation of 'Cheetah' image.

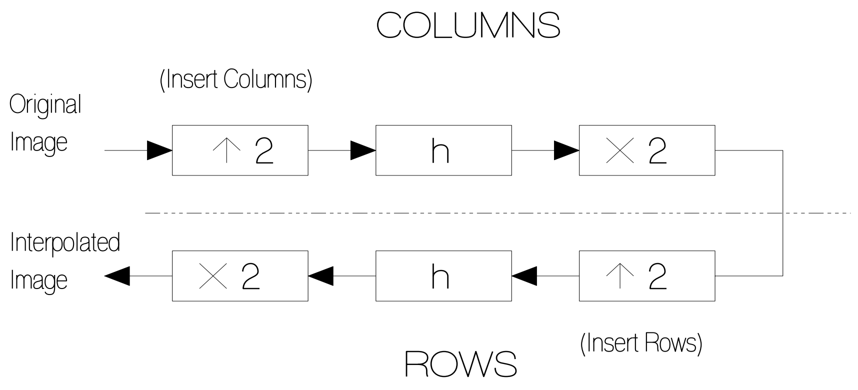

Figure 6.5: Subband interpolation of images. h is the low pass Quadrature Mirror Filter.

interpolation, no artifacts are introduced.

Chapter 7

Conclusions

In this thesis, some new methods of improving the efficacy of fractal image compression have been explored. The aim is to develop new fractal coding schemes, which, for a given compression ratio, yield either a significant reduction in complexity at the cost of a marginal reduction in quality, or a significant improvement in quality for a marginal increase in computation. All comparisons are with reference to Jacquin's block matching technique [8] which involves an exhaustive search in the domain library or a sub-library.

In Chapter 3 we introduced the method of block matching through inner product, which is twice as fast as the conventional least squares method of block matching. In Chapter 4 we introduced a new technique of FFT-based block matching, in which the set of isometric transformations chosen for each domain block were different from the regular 8 transformations consisting of rotations and reflections. The new set of transformations consisting of circular shifts of the pixels of a shrunken domain block expressed as a vector, facilitate the use of a single FFT for computing the inner product of a range block with the entire set of codebook blocks generated by a domain block. Since the number of codebook blocks per domain block is much larger than 8, the size of the domain library required for a given number of codebook blocks gets correspondingly reduced. Thus, for a given compression ratio, the new method yields a very significant increase in the computational speed as compared to Jacquin's method, for the cost of a slight degradation in the decoded image quality. For example, for range block sizes of 8×8, the FFT-based block matching method is approximately 25 times faster than Jacquin's method and has a PSNR approximately 0.75 dB lower. The reduction in PSNR indicates that some codebook blocks generated by the new transformation algorithm are not useful. The new method can be speeded up further by restricting the FFT-based matching of each range block to only a small subset of domain blocks whose magnitude DFT's are relatively closer to that of the range block. The reduction in computation achieved with this restriction is considerable, with negligible additional loss in image quality.

We have also attempted a hybrid scheme wherein the above block matching technique is used to encode the error image resulting from the first stage of a low-bit-rate fractal compression scheme. For encoding the errors we choose the domain blocks from a highly decimated image. We have found that we need up to three levels of approximations for many range blocks for coding the residual error in a block, within some tolerance.

In Chapter 5 we introduced a new family of fast, unitary, feature highlighting transforms (FHT). The FHT is made to highlight features by choosing the basis vectors of the transform as an impulse response of an all-pass filter, closest to the feature we desire to highlight, and all its translates. The FHT was used to significantly improve the quality of the edges of decoded images of a low bit rate fractal compression scheme. We have also briefly explored some modifications of the FHT where tradeoffs are made between the spatial / temporal resolution of the basis vectors and their closeness to the feature we desire to highlight. If the resolution is further reduced (say, by making the basis vectors orthogonal to every fourth shift) we could highlight multiple features. Other uses of FHT and its variants are yet to be investigated.

The IFS code naturally lends itself to interpolation since decoding can be done at any image resolution. In order to get interpolated images of good quality, the quality of the non-interpolated decoded images should be good, which may be achieved by choosing range blocks of small size. However, our studies indicate that fractal interpolation does not work well with range blocks of small size. Hence, we have devised an IFS coding technique of improving the quality of the non-interpolated images without reducing the size of the range blocks. In this technique, multiple domain are used to approximate each range block. Though IFS interpolation yields, in general, sharper pictures than other methods, it also introduces a lot of artifacts which are, at times, unacceptable. Moreover the improvement in the quality of interpolated images with increase in the bit rate of the IFS code is only marginal. Still, the fractal interpolation scheme can be useful as a low-complexity, low-bit-rate, compression scheme, where the original image is decimated, IFS coded, and then decoded at a higher resolution. For this scheme too, by enhancing the edges of the image using the FHT before IFS coding, the quality of the decoded image can be significantly improved. We have compared the quality IFS interpolation with linear interpolation, interpolation using DCT, and subband interpolation. We see that the the most acceptable quality of interpolation is achieved through subband interpolation. The sharpness of subband interpolated images is better than that of linear interpolation or DCT, and, unlike IFS interpolation, no annoying artifacts are added.

Appendix A

Original Images Used

Figure A.1: 128×128 images. ‘Elephants’, ‘Squirrels’, ‘Girl’, and ‘Cheetah’.

Figure A.2: 512×512 images. ‘Lena’, ‘Peppers’,‘Boats’ and ‘Baboon’.

Bibliography

[1] N.S.Jayant and P.Noll, *Digital Coding of Waveforms*, Englewood Cliffs, NJ, Prentice Hall, 1984.

[2] N.Ahmed, T.Natarajan, K.R.Rao, 'Discrete Cosine Transform', *IEEE Trans. Comput.* C-**23**, pp 90-93, Jan 1974.

[3] G.K.Wallace, 'The JPEG Still Picture Compression Standard', *Commun. ACM*, **34**, pp 31-44, Apr 91.

[4] I.Daubechies, 'The Wavelet Transform, Time-Frequency Localization and Signal Analysis', *IEEE Trans. on IT*, **36**(5), pp 961-1005, Sep 90.

[5] J.M.Shapiro, 'Embedded Image Coding Using Zero-Trees of Wavelet Coefficients', *IEEE Trans. on Signal Proc.*, **41**(12), pp 3445-62, Dec 93.

[6] J.W.Woods and S.D.O'Neil, 'Subband Coding of Images', *IEEE Trans. of ASSP*, **34**(5), Oct 86.

[7] P.J.Burt and E.H.Adelson, 'The Laplacian Pyramid as an Efficient Image Code', *IEEE Trans. on Commn.*, pp 532-540, Apr 83.

[8] A.E.Jacquin, 'Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations', *IEEE Trans. on Image Proc.*, **1**(1), pp 18-30, Jan 1992.

[9] A.E.Jacquin, 'Fractal Image Coding: A Review', *Proc. of the IEEE*, **81**(10), pp 1451-65.

[10] M.Barnsley, *Fractals Everywhere*, Academic Press, San Diego, CA, 1988.

[11] M.Barnsley and A.D.Sloan, 'A Better Way to Compress Images', *Byte*, pp 215-23, Jan 1988.

[12] Y.Fisher, ‘Mathematical Background’, in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 2, Springer Verlag, NY, 1995.

[13] Y.Fisher, E.W.Jacobs, R.D.Boss, ‘Iterated Transform Image Compression’, NOSC TR-1048, Naval Ocean Systems Center, San Diego, CA.

[14] E.W.Jacobs, Y.Fisher and R.D.Boss,‘Image Compression: A study of the Iterated Transform Method’, *Signal Processing*, **19**, pp 251-63, 1992.

[15] Y.Fisher, ‘Fractal Image Compression with Quad-trees’ in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 3, Springer Verlag, NY, 1995.

[16] Y.Fisher, ‘Introduction’, in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 1, Springer Verlag, NY, 1995.

[17] D.M.Munro and F.Dudbridge, ‘Fractal Approximation of Image Blocks’, ICASSP-92, III, pp 485-88

[18] F.Dudbridge, ‘Least Squares Block Coding by Fractal Functions’, in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 3, Springer Verlag, NY, 1995.

[19] D.M.Munro, ‘A Hybrid Fractal Transform’, ICASSP-93, V, pp 169-72.

[20] D.M.Munro and S.J.Woolley, ‘Fractal Image Compression Without Searching’, ICASSP-94, V, pp 557-60

[21] S.J.Woolley and D.M.Munro, ‘Optimum Parameters for Hybrid Fractal Image Coding’, ICASSP-95, IV, pp 2571-74.

[22] H.Zhang, X.Gao and Z.He, ‘A Modified Fractal Transform’, ICASSP-95, IV, pp 2567-70.

[23] G.E.Oien, S.Lepsoy and T.A.Ramstad, ‘An Inner Product Space Approach to Image Compression by Contractive Transformations’, ICASSP-91, IV, pp 2773-76.

[24] G.Vines, ‘Orthogonal Basis IFS’, in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 3, Springer Verlag, NY, 1995.

[25] D.Saupe and R.Hamzaoui, 'Complexity Reduction Methods for Fractal Image Compression' *IMA Conf. in Image Proc. : Mathematical Methods and Applications*, T.M.Blacklegde (ed), Oxford University Press, 1995.

[26] D.Saupe, 'Breaking the Time Complexity of Fractal Image Compression', Technical Report 53, Institut fur Informatik, Universitat Frieburg, 1994.

[27] K.W.Barthel, T.Voye and P.Noll, 'Improved Fractal Image Coding', Proceedings of the International Picture Coding Symposium, PCS-93, Section 1.5, March 93.

[28] B.Ramamoorthi and A.Gersho, 'Classified Vector Quantization of Images', *IEEE Trans. Commn.* COM-**34**, 1986.

[29] R.D.Boss and E.W.Jacobs, 'Archetype Classification in an Iterated Transform Image Compression Algorithm' in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 4, Springer Verlag, NY, 1995.

[30] S.Lepsoy and G.E.Oien, 'Fast Attractor Image Coding by Adaptive Codebook Clustering' in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 9, Springer Verlag, NY, 1995.

[31] T.Bedford, F.M.Dekking and M.S.Keane, 'Fractal Image Coding Techniques and Contractive Operators', *Nieuw Arch. Wisk* (4) **10**,3, pp 185-218, 1992.

[32] J.Kominek, 'Algorithm for Fast Fractal Image Compression', Proc. of SPIE Digital Video Compression: Algorithms and Technologies, **2419**, pp 296-305, 1995.

[33] C.Frigaard, J.Gade, T.Hemmingsen and T.Sand, 'Image Compression Based on Fractal Theory', Technical Report, Institute for Electronic Systems, Aalberg University, Denmark, 1994.

[34] B.Bani-Eqbal, 'Speeding Up Fractal Image Compression', Proc. of SPIE: Still Image Compression, **2418**, pp 67-74, 1995.

[35] B.E.Wohlberg, G.de Jager, 'Fast Image Domain Fractal Compression by DCT Domain Block Matching', *Elect. Lett.* **30**, pp 474-75, Mar 94.

[36] J.Kominek, 'Convergence of Fractal Encoded Images' Proc. of Data Compression Conference, pp 242-251, 1995.

[37] L.Lundhiem, 'A Discrete Framework for Fractal Signal Modeling', in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 7, Springer Verlag, NY, 1995.

[38] B.Hurtgen, 'Contractivity of Fractal Transforms for Image Coding', *Elect. Lett.*, **29**(20) pp 1749-50.

[39] J.Domaszenicz and J.Vaishampayan, 'Graph Theoretical Analysis of Fractal Transforms', ICASSP-95, IV, pp 2559-62

[40] G.E.Oien and S.Lepsoy, 'A Class of Fractal Image Coders With Fast Decoder Convergence', in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 8, Springer Verlag, NY, 1995.

[41] Z.Baharav, D.Malah and E.Karnin, 'Hierarchical Interpretation of Fractal Image Compression and its Applications', in Y.Fisher (ed), *Fractal Image Compression: Theory and Application*, Chapter 5, Springer Verlag, NY, 1995.

[42] A.Bogdan,'Multiscale Fractal Image Coding and the Two-scale Difference Equation', Columbia University Technical Report, TR-358-94-05, pp 1-14, 1994.

[43] H.Krupnik, D.Malah and E.Karnin,'Fractal Representation of Images via the Discrete Wavelet Transform', IEEE 18th Conv. in Isreal, Tel-Aviv, Mar 1995.

[44] R.Rinaldo and G.Calvagno, 'An Image Coding Scheme Using Block Prediction of the Pyramid Subband Decomposition', IEEE International Conference on Image Processing, ICIP-94, 1994.

[45] S.G.Mallat, 'A Theory for Multiresolution Signal Decomposition: The Wavelet Representation.' *IEEE Trans. of PAMI*, **11**(7), pp 674-693, Jul 89.

[46] G.Oien, 'Parameter Quantization in Fractal Image Coding', ICASSP-94, V, pp 142-146

[47] A.R.Butz, 'Alternative Algorithm for Hilbert's Space Filling Curve',*IEEE Trans. on Computers*, pp 924-26, Apr 1971.

[48] P.Duhamel and M.Vetterli, 'Improved Fourier and Hartley Transform Algorithms: Application to Cyclic Convolution of Real Data.' *IEEE Trans. Acoust., Speech and Signal Processing*, vol **ASSP-35**, pp 818-824, June 1987.

[49] P.Duhamel, 'Implementation of "Split-Radix" FFT Algorithms for complex, real and real-symmetric data,' *IEEE Trans. on Acoust., Speech, Signal Processing*, vol-**ASSP-34**, pp 285-295, Apr. 1986.

[50] P.Duhamel, 'Algorithms Meeting the Lower Bounds on the Multiplicative Complexity of Length - 2^n DFT's and Their Connection with Practical Applications,' *IEEE Trans. Acoust., Speech, Signal Processing*, vol **38**, pp 1504-1511, Sep. 1990.

[51] H.S.Malvar,*Signal Processing with Lapped Transforms*, Artech House Inc.,Norwood, 1992.

[52] P.A.Regalia, S.K.Mitra and P.P.Vaidyanathan, 'The Digital All Pass Filter: a Versatile Signal Processing Building Block', *Proc. IEEE*, Vol **76**, Jan 1988.

[53] P.P.Vaidyanathan, *Multirate Systems and Filter Banks*, Englewood Cliffs, N.J, Prentice Hall Signal Processing Series, 1994.

[54] M. Vetterli and J Kovacevic, *Wavelets and Subband Coding*, Englewood Cliffs, N.J, Prentice Hall Signal Processing Series, 1995.

[55] K.Popat and K.Zeger, 'Robust Quantization of Memoryless Sources Using Dispersive FIR Filters', *IEEE Trans. on Commn.*, Vol **40**, No 11, November 1992.

[56] H.W.Strube, 'How to Make an All-Pass Filter with a Desired Impulse Response', *IEEE Trans. on ASSP*, ASSP-**30**, No 2, April 1982.

[57] I Daubechies, 'Orthonormal Bases of Compactly Supported Wavelets',*Commun. of Pure and Appl. Math.* **41**, pp 909-996, Nov 1988.