A FRACTAL-BASED IMAGE BLOCK-CODING ALGORITHM

Mohammad Gharavi-Alkhansari and Thomas S. Huang

Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
405 N. Mathews Ave., Urbana, IL 61801 U.S.A.

ABSTRACT

This paper presents a new block-coding algorithm for grey-scale images based on a fractal approximation. Image blocks are approximated by a linear combination of a set of almost orthogonal basis blocks made up of, (i) a set of simple image-independent blocks, and (ii) a set of image-*dependent* blocks generated from transformed blocks of bigger size in the same image. To find the approximation for each block, the block is projected onto the space spanned by the set of basis blocks. For each block, the biggest coefficients plus the index of the corresponding basis blocks and the number of basis blocks used make the code. The number of basis vectors used for coding each block is the number of basis blocks needed to approximate the block within a given error limit. We also present the results of a study on the effect of variations on Jacquin's fractal-based image coding algorithm.

1. INTRODUCTION

The method described in this paper has its roots in the work by Barnsley on image compression, and later on by Jacquin [1] [2]. In Jacquin's algorithm, an image is partitioned into blocks called range blocks, which are classified by their complexity. Simple range blocks are coded by their average value and more complex ones are approximated by a single transformed block of a bigger size called domain block, which has to be of the same class in the same image. For complex range blocks, the parameters of transformations and the address of the domain blocks are used as the code for the range blocks. Then each range block is split into four smaller subblocks and the error in encoding each subblock is computed. If the error is higher than a threshold, the smaller blocks are encoded separately. Later on, different variations of the Jacquin's algorithm were used for image coding in which linear and/or quadratic components of the pixel values in a block are treated differently [3][4]. We studied other variations in Jacquin's algorithm and used different range block sizes, different domain block sizes, and different numbers of domain blocks. Our studies showed that for this algorithm, when implemented without any block splitting,

This research was supported by a grant from Mitsubishi Electric.

1. The error involved in coding each range block has strong positive correlation with the variance of the pixel values in the block.
2. The error increases with increasing the shrinking factor. This means that range blocks tend to find better matches in domain blocks that are close to range blocks in size. In other words, self similarity in a natural image decreases as the difference in scale increases and domain blocks had usually better be chosen from scales of smallest difference with the range blocks. But on the other hand decreasing the size of the domain blocks excessively, reduces the contractivity of the transformation and causes more error in the decoding process. In our new process described in the next section, the size of domain blocks is chosen to be twice the size of the range blocks.
3. Allowing more rotations and shrinking factors or more domain blocks does not improve the performance of the algorithm significantly.

Barnsley and Jacquin also showed how the information about the relation between smaller and larger parts in an image can be used to reconstruct the image by an iterative decoding process [1] [2]. The reconstruction is done by iteratively applying a contractive transformation on an initial image, and the convergence is assured by the contraction mapping theorem.

2. ENCODING PROCESS

Our new method is, in a sense, a generalization of the methods proposed by Jacquin and Øien et al. [2] [3]. In the encoding process of our method, an image is first tiled with square range blocks of size $S_R \times S_R$. Each range block can be considered as a N-dimensional vector, denoted by $\vec{r}_j, j = 1, \cdots, N_R$, where $N = S_R \times S_R$, and N_R is the total number of range blocks. Then a set of normalized almost orthogonal N-dimensional basis vectors, called basis blocks, are constructed. This set is made up of two subsets, (i) a set of image-independent basis blocks, called fixed basis blocks (FBB), and (ii) a set of image-dependent basis blocks (IDBB). These sets are constructed in such a way that, on the average, each range block can be closely approximated by a linear combination of a small number of basis blocks. Then in the encoder, the index of the basis blocks and their corresponding coefficients make the code

for each range block. The basis blocks are near orthonormal. The construction of the FBBs is done by the encoder designer and is done once for the class of all the images to be encoded. This construction is based on our general knowledge of the nature of the image blocks. The relation between these basis blocks and the range blocks in the image is not based on the fractal properties of the image. On the other hand, the construction of the IDBBs is done for each image separately. In this construction, the encoder tries to find blocks of size $S_D \times S_D$ which can strongly contribute to the construction of range blocks. The assumption is that the residuals of the range blocks, i.e. range blocks after removal of the FBB components, can be efficiently encoded by a linear combination of these IDBBs. This assumption is based on one interpretation of fractal property of natural images that for sections of an image, strong components exist at some bigger sections of the same image (self similarity at different scales).

2.1. Fixed Basis Blocks

For FBBs, a set of $N_1 < N$ orthonormal blocks $\vec{u}_i, i = 1, \cdots, N_1$, are defined. We chose $N_1 = 6$, and defined the FBBs basicly of the following form (before normalization):

$$\begin{array}{lll} z = 1, & z = x, & z = y, \\ z = x^2 - a, & z = y^2 - a, & z = xy. \end{array}$$

The z variable represents the pixel value and the origin of the x and y coordinates is the center of the block. x axis is vertical and downward, and y axis is horizontal and towards right. a is a constant and its value depends on the size of the block. A linear combination of these six blocks can generate any quadratic block of the form $z = a_1 + a_2x + a_3y + a_4x^2 + a_5y^2 + a_6xy$.

Other kinds and numbers of FBBs may be used depending on the nature of the images to be encoded. The general guideline is that range blocks in the set of images to be encoded must have strong components of FBBs. Our choice of FBBs seems reasonable for most natural images which have a lot of shades and strong correlation between neighboring pixel values.

2.2. Image-Dependent Basis Blocks

A set of N_2 IDBBs $\vec{v}_i, i = 1, \cdots, N_2$, are constructed such that

$$\begin{array}{lll} \|\vec{v}_i\| = 1 & \text{for} & 1 \le i \le N_2, \\ \left|\langle \vec{v}_i, \vec{v}_j \rangle\right| \le t_c & \text{for} & 1 \le i, j \le N_2, \quad i \ne j, \\ \left|\langle \vec{v}_i, \vec{u}_j \rangle\right| \le t_c & \text{for} & 1 \le i \le N_2, \quad 1 \le j \le N_1, \end{array}$$

where t_c is the maximum allowed deviation from zero for the cosine of the angle between two IDBBs or an IDBB and a FBB. t_c is a parameter of the encoding process and is chosen small enough to give near-orthogonal properties to our basis blocks but big enough to allow enough number of IDBBs to be selected in the process of finding the IDBBs as described below. IDBBs are generated by the following procedure,

1. A pool of domain blocks of size $S_D \times S_D$ taken from the image is made. One may choose to use all possible blocks of size $S_D \times S_D$ in the image. We denote these domain blocks by $\vec{d}_n, n = 1, \cdots, N_D$, where N_D is the number of domain blocks used. For the results of this paper, S_D is chosen equal to $2S_R$, although the ratio S_D/S_R can have any other value as long as it is greater than one (see previous section).
2. The pool is expanded by also including simply transformed versions of the domain blocks which are made by reflecting them against their horizontal and vertical axis and rotating them by multiples of 90 degrees [2]. This gives us a total of $N_T = 8$ different versions for each domain block. The members of this expanded pool are denoted by $\vec{f}_k, k = 1, \cdots, N_D N_T$.
3. Each $\vec{f}_k$ is shrunk by a factor of S_D/S_R in each direction.
4. Component of the same form as any of the N_1 FBBs are removed from $\vec{f}_k$'s. The resulting blocks are denoted by $\vec{g}_k, k = 1, \cdots, N_D N_T$.
5. Each $\vec{g}_k$ is given a priority number p_k defined as

$$p_k = \sum_{j=1}^{N_R} \left[min\left(\frac{|\langle \vec{r}_j, \vec{g}_k \rangle|}{\|\vec{g}_k\|}, \|\vec{g}_k\| \right) \right]^2, k = 1, \cdots, N_D N_T.$$

   p_k represents the overall contribution of $\vec{g}_k$ to the construction of all range blocks. For each $\vec{g}_k$ the contribution to each range block is limited to the norm of $\vec{g}_k$ to avoid giving high priority numbers to $\vec{g}_k$'s of small norms. This is to keep the IDBB-generating transformations, applied on domain blocks, contractive.
6. $\vec{g}_k$'s are sorted in descending order of their priority numbers.
7. The value of t_c is selected.
8. The $\vec{g}_k$ with the highest priority number is selected as the first IDBB.
9. The following IDBBs are chosen to be the $\vec{g}_k$'s with highest priority numbers whose angles with all the previously chosen IDBBs are close to 90 degrees within the allowable threshold, determined by t_c.
10. If the number of IDBBs becomes greater than $(N - N_1)$ only the first $(N - N_1)$ ones are selected.
11. All the basis blocks are normalized, giving $\vec{v}_i, i = 1, \cdots, N_2$.

For the IDBB $\vec{v}_i$, we denote the index of the originating domain block by n_i and the transformation that converts $\vec{d}_{n_i}$ to $\vec{v}_i$ (a combination of shrinking, rotation, reflection, removal of FBB components, and normalization) by $\mathcal{T}_i$, i.e., $\vec{v}_i = \mathcal{T}_i(\vec{d}_{n_i})$. The addresses of $\vec{d}_{n_i}, i = 1, \cdots, N_2$ and the parameters of $\mathcal{T}_i, i = 1, \cdots, N_2$ are sent as an overhead for each image. The overhead also includes the maximum and the minimum pixel values of the encoded image. It must be noted that the pixel values of the IDBBs are neither sent nor needed for the decoding process.

2.3. Code for Range Blocks

The process of making basis blocks, gives us a total of $N_B \stackrel{\text{def}}{=} N_1 + N_2$ basis blocks where $N_B \le N$. For simplicity

of notation, we rename the ordered basis blocks $(\vec{u}_1, \cdots, \vec{u}_{N_1}, \vec{v}_1, \cdots, \vec{v}_{N_2})$ by $(\vec{b}_1, \cdots, \vec{b}_{N_B})$. After the basis blocks are found, each range block $\vec{r}_j$ is projected onto the space spanned by the N_B basis blocks $\vec{b}_i, i = 1, \cdots, N_B$, by standard least squares methods, giving coefficients $\alpha_{j,i}, i = 1, \cdots, N_B$. For each range block, these coefficients are sorted in descending order of their absolute value. We denote the second index (subscript) of these sorted coefficients by $I_{j,m}, m = 1, \cdots, N_B$. Beginning with the coefficient $\alpha_{j,I_{j,1}}$ which has the largest absolute value, we multiply these coefficients by their corresponding basis blocks and add them together. After the L-th accumulation, we have $\vec{\hat{r}}_{j,L} = \sum_{m=1}^{L} \alpha_{j,I_{j,m}} \vec{b}_{I_{j,m}}$. For each $\vec{r}_j$ this accumulation is continued, giving a non-increasing error $e_j(L) = \|\vec{\hat{r}}_{j,L} - \vec{r}_j\|$. This accumulation is stopped if any of the following conditions are satisfied:

1. $e_j(L) \leq E$ where E is an error threshold set by the designer of the encoder.
2. $e_j(L-1) - e_j(L) \leq F$ where F is a threshold for the rate of decrease of the error, set by the designer of the encoder. This condition is usually satisfied when $N_B < N$ and the range block can not be efficiently approximated by the selected basis blocks.
3. $L = N_B$.

For the range block $\vec{r}_j$, we denote the value of L at which the above accumulation stops, by M_j. The number M_j, the coefficients $\alpha_{j,I_{j,m}}, m = 1, \cdots, M_j$, and the indices of the basis blocks corresponding to these coefficients: $I_{j,m}, m = 1, \cdots, M_j$, make the code for each range block $\vec{r}_j$:

$$\vec{r}_j \xrightarrow{\text{code}} M_j, (\alpha_{j,I_{j,m}}, I_{j,m}, m = 1, \cdots, M_j)$$

We have not applied any quantization to the coefficients in the code. The results presented in the last section of this paper are based on these non-quantized values.

3. DECODING PROCESS

Our decoding process is an iterative process similar to the one suggested by Jacquin [2]. We can begin the decoding with any initial image, e.g. an image with all its pixel values equal to zero. However, the decoding process becomes faster when the initial image is constructed from the coefficients of each range block corresponding to the N_1 FBBs. At K-th iteration, we make a set of basis blocks $\vec{\hat{b}}_{i,K}, i = 1, \cdots, N_B$. For $i = 1, \cdots, N_1$, $\vec{\hat{b}}_{i,K} = \vec{b}_i$, but for $i = N_1 + 1, \cdots, N_B$, we have to make an approximation for the IDBBs, namely $\vec{\hat{b}}_{i,K} = \mathcal{T}_i(\vec{\hat{d}}_{n_i,K})$ where $\vec{\hat{d}}_{n_i,K}$ is the block of size $S_D \times S_D$ with the same address as $\vec{d}_{n_i}$, in the image resulting from iteration $K-1$. Now having the FBBs and an approximation of IDBBs, we can reconstruct an approximation $\vec{\hat{r}}_{j,K}$ of the range block $\vec{r}_j$ by applying,

$$\vec{\hat{r}}_{j,K} = \sum_{m=1}^{M_j} \alpha_{j,I_{j,m}} \vec{\hat{b}}_{I_{j,m},K}.$$

The pixel values in the range block $\vec{r}_j$ are then limited to values between the minimum and the maximum intensities of the encoded image (known to the decoder from the overhead in the code). This procedure is repeated until the change in the images are small. Although the convergence of this decoding process is not mathematically proven, in practice the decoding process converges after a few iterations for all of our natural test images.

Table 1: Paremeters of the encoder.

size of image	256×256
S_R	8
S_D	16
N_D	58081
N_T	8
N_1	6
t_c	0.16
E/S_R	6.
F/S_R	1.

4. COMPARISON

Regarding the encoder, the main differences between this algorithm and the one proposed by Jacquin are as follows:

1. For each range block a linear combination of a set of basis blocks is used instead of only one domain block.
2. In addition to using IDBBs, N_1 FBBs are also used. In Jacquin's method only the information of the average of the range blocks is transmitted.
3. No classification of range blocks and no block splitting is done in this method. More complex blocks usually need a higher number of basis blocks for coding.

For the decoder, in addition to the encoder-related differences, our method is different from Jacquin's method in the sense that instead of beginning with an arbitrary initial image, we begin with an image which is close to the desired image by using the coefficients of the FBBs. This makes the decoding process one step shorter.

5. TEST RESULTS

Our algorithm was applied to the 8-bit/pixel test image "Lena". The parameters of the encoding process are given in Table 1. The encoder found $N_2 = 26$ IDBBs. The average number of basis blocks used for encoding each range block is 3.71. The estimated rms error at the encoding process is 9.79 (PSNR=28.3 dB). Figure 1 shows the number of times each basis block $\vec{b}_i, i = 1, \cdots, N_B$, was used for encoding range blocks. It can be seen that the FBBs are more extensively used than the IDBBS. Figure 1 also shows modified priority numbers $\bar{p}_k \stackrel{\text{def}}{=} (p_k/N_R)^{\frac{1}{2}}, k = 1, \cdots, N_D N_T$, and the modified norms $(\|\vec{g}_k\|/S_R)$ of the $\vec{g}_k$'s corresponding to IDBBs. It seems that the priority numbers are a good measure of how extensively each IDBB is used. Figure 2 shows the original test image "Lena". The initial image and the images resulting from the iterative decoding process after the first and second iterations are shown

Figure 1: (a) Number of times each basis block $\vec{b}_{n_i}, i = 1, \cdots, N_B$, was used for encoding range blocks, (b) modified priority number of the $\vec{g}_k$'s corresponding to IDBBs, (c) the modified norms of the $\vec{g}_k$'s corresponding to IDBBs.

in Figure 3. The error of the decoded images remains almost constant after two iterations and the final rms error of 9.86 (PSNR=28.3 dB) is very close to the rms error of 9.79 estimated in the encoding process.

6. REFERENCES

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

[2] A. E. Jacquin, "A novel fractal block-coding technique for digital images," *Proc. IEEE ICASSP*, pp. 2225–2228, 1990.

[3] G. E. Øien, S. Lepsøy, and T. A. Ramstad, "An inner product space approach to image coding by contractive transformations," *Proc. IEEE ICASSP*, pp. 2773–2776, 1991.

[4] D. M. Monro, and F. Dudbridge, "Fractal approximation of image blocks," *Proc. IEEE ICASSP*, Vol. 3, pp. 485–488, 1992.

Figure 2: Original test image "Lena"

Figure 3: (a) Initial image made by using fixed basis blocks (PSNR=25.6 dB), (b) decoded image after one iteration (PSNR=28.0 dB), (c) decoded image after two iterations (PSNR=28.3 dB).