DOI: 10.21917/ijivp.2010.0018

ENHANCED ITERATION-FREE FRACTAL IMAGE CODING ALGORITHM WITH EFFICIENT SEARCH AND STORAGE SPACE

A.R. Nadira Banu Kamal[1] and S. Thamarai Selvi[2]

[1]*Department of Computer Science, TBAK College for Women, Tamil Nadu, India*
E-mail: nadirakamal@gmail.com

[2]*Department of Computer Technology, MIT Campus, Anna University, Tamil Nadu, India*
E-mail: stselvi@annauniv.edu

Abstract

An enhanced iteration free fractal algorithm is proposed in this research paper to design an efficient domain pool for image compression. The proposed methodology reduces the coding process time, intensive computation tasks and also the memory requirements. The redundancies in the domain pool are reduced by the Linde Buzo Gray (LBG) Algorithm. For each range block, vector features such as mean value, edge strength, and texture strength are used to delete the irrelevant domain block. A pruning condition for terminating the searching process to find the best domain block from the domain pool is used. The codes are stored efficiently by comparing the values of the previous coded range blocks. The performance of the proposed method is compared with the existing iteration free fractal code for the benchmark images on the parameters like coding time, memory capacity and image quality. From the results of the computer simulation, the proposed method achieves excellent performance in coding time. The enhancement scheme for iteration free fractal image coding using vector quantization resulted in a reduction of 5.7 times and 11.5 times than the existing iteration free fractal code method for the single block partition of size 8x8 and 4x4 respectively on the Lena image for the codebook of size 16. The reduction in time is still higher in using code books of higher levels.

Keywords:

Block Average, Domain Pool, Fractal Image Compression, Iteration Free Fractal Code, LBG Algorithm

1. INTRODUCTION

The fractal coding scheme is a technique for image compression [1]-[5]. It has become one of the most popular modern image coding methods in recent years. There are several different ways in which image files can be compressed. For Internet use, the two most common compressed graphic image formats are the JPEG format and the Graphics Interchange Format (GIF) format. JPEG compression is quite effective at low or moderate compression ratio up to 20 to 1. Beyond this the image becomes very blocky as the compression increases the image quality is too poor for practical use. Other techniques for image compression include the use of fractals.

This method has not gained widespread acceptance for use on the Internet. However, it is being explored because it offers higher compression ratios for lower bit rates than the JPEG method [8]. Fractal image coding has many advantages, such as the high quality at compression ratio. The cause of fractal image coding with high compression is that it uses the feature of self similarity. However, fractal encoding has a fatal drawback of consuming more time during its encoding process. By overcoming this limitation, fractal image coding can be widely applied.

In the fractal coding schemes, an image is partitioned into non overlapping range blocks. The larger domain blocks D are selected from the same image which can overlap. A grayscale image is encoded by mapping the domain block D to the range block R with contractive affine transformation [2] given by (1),

$$\hat{R} = i\{\alpha .(SoD) + \Delta g\} \quad (1)$$

where the operation S o D represents the contraction that maps the domain block D to a range block R. The parameters (called the *fractal code*) describing the contractive affine transformation, which has the minimum matching error between the original range block R and the coded range block $\hat{R}$, are transmitted or stored. The fractal code consists of the contrast scaling α, luminance shift Δg or the block mean (the average pixel value of the range block) μ_R, isometry *i*, and the position P_D of the best-match domain block in the domain pool. In the decoding stage, an arbitrary image is given as the initial image and the decoded image is recursively reconstructed by applying the contractive affine transformation to the iterated image.

The domain pool in fractal coding schemes consists of the domain blocks obtained by sub-sampling the original image, or choosing the neighboring blocks of the range block. Generally, a better coding performance is achieved when a larger domain pool is used in the encoding stage. However, there exist some redundancies between the domain blocks, especially for a large domain pool or the domain blocks chosen from the neighboring blocks of the range block. By reducing such redundancies between the domain blocks, the constructed domain pool is encoded efficiently resulting in reduced time for decoding an image of good quality. The Linde Buzo Gray (LBG) algorithm [6] used to generate the codebook in the Vector Quantization (VQ) techniques has the ability to reduce the redundancies between the training vectors. Using the same codebook in both the encoder and decoder, we can encode/decode an image [7]. Since there is no transmission of domain blocks in fractal coding schemes, the LBG algorithm cannot be directly applied to generate the domain blocks. In order to obtain the same domain blocks in both the encoder and decoder in the fractal coding scheme, an iteration-free fractal coding scheme was proposed [8] using synthetic code book. This method is improved by reducing the domain pool for each range block which results in efficient coding time in the proposed method.

The block mean of each range block is found in the modified contractive affine transformation of the fractal codes. Hence the same mean image, whose pixel values are the block means of all the range blocks, can be generated in both the encoder and the decoder. The LBG algorithm is applied to design the domain pool using the mean image. The LBG method reduces the

redundancies between the generated domain blocks and thus the constructed domain pool is efficient compared to the fractal schemes using iterations. The coding performance is further improved in the proposed algorithm by extracting the vector features and a condition for terminating the searching process to find the best matching domain block of a range block. This helps in reducing the distortion calculations to find the best match for the range block. Extra computations for these additional conditions are very small and have reduced the coding time to a great extent. The fractal codes obtained are stored compactly by comparing them with the previous codes. The computer simulation shows that a high reduction in coding, a good reduction in memory size and acceptable quality of decoded image is obtained. This proposed algorithm is simple and suitable for hardware implementation.

The rest of the paper is organized as follows. In Section 2 the architecture of the proposed enhanced iteration free fractal coding method is presented. Section 3 explains the algorithm for the proposed method. Section 4 explains the implementation of the enhanced iteration free fractal coding method. Results and performance comparisons are shown and discussed in Section 5, followed by the merits and applications of the proposed method in Section 6. In the final section conclusions and suggestions are offered.

2. ARCHITECTURE OF ENHANCED ITERATION-FREE FRACTAL IMAGE CODING

In order to obtain the same domain blocks in both the encoder and decoder without using an off-line transmission, we use the mean information of the range blocks that are hidden in the fractal codes. The LBG algorithm and vector features are applied to reduce the redundancies between the generated domain blocks in the domain pool. The architecture of the proposed method is described in Fig.1 (a) and (b).

In the preprocessing stage, the input M x N image under coding is divided into non overlapping square blocks of B x B pixels called the range blocks. Then the mean and variance of each range blocks are determined. After the mean of all the range blocks are obtained, a mean image of size M/B x N/B with each pixel corresponding to the block mean is generated. The mean image must be larger than the size of the range block i.e. M/B x N/B > B x B. Otherwise it will not be easy to find a good mapping between the domain and range blocks because only a few domain blocks can be taken from the mean image.

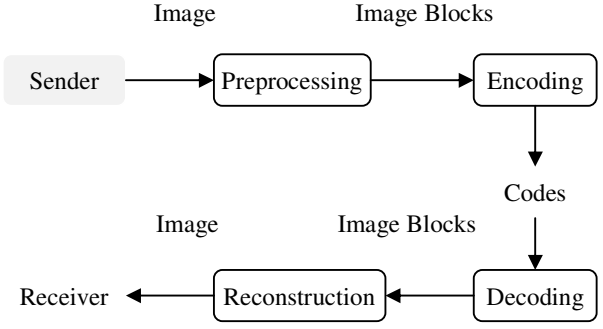


Fig.1(a). Architecture of the proposed method

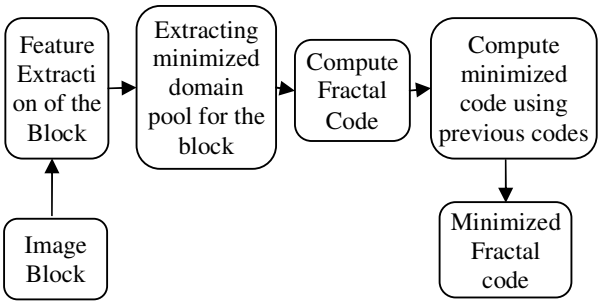


Fig.1(b). Architecture of the Encoder

The initial domain pool is generated using the mean image and the redundant domain blocks are eliminated using the LBG algorithm. In the encoder if the variance of the range block is smaller than the threshold value *E*, then the range block is coded by the mean. Otherwise, the range block will be coded by the contractive affine transformation. The aim of the proposed scheme is to find for each image block the domain block and the transformation parameters that minimize the distortion between the image block and the transformed domain block in a minimized time. In our proposed method the number of calculations to determine this is reduced by extracting the features of the range block like mean, edge strength and texture strength and comparing it with the domain pool and eliminates redundant domain blocks. The transformations are applied only to these domain blocks and the transformation parameters that minimize the distortion between the image block and the transformed domain block is coded. This code is further compared with the previous codes and the minimized fractal code is determined.

In the decoder, the mean information of each range block is extracted from the minimized fractal codes. Using this information the mean image is constructed. The domain pool is obtained using the LBG algorithm. The image is decoded block by block by applying the transformation parameters to the domain block as per the code.

3. ALGORITHMS FOR ENCODING AND DECODING

3.1 ENCODER

The basic flow chart of the encoder in the proposed enhanced iteration-free fractal code scheme is shown in Fig. 2. The input image is partitioned into blocks. The mean and variance of each block is calculated. The initial domain pool is generated using the mean image and the redundant domain blocks are eliminated using the LBG algorithm. In the encoder if the variance V{R} of the range block

$$V\{R\} = \frac{1}{B^2} \sum_{0 \le i,j < B} (r_{i,j} - \mu_R)^2 \qquad (2)$$

(where $r_{i,j}$ denotes the (i, j)th pixel in the range block of size BxB) is smaller than the threshold value *E*, then the range block is coded by the mean. Otherwise, the range block will be coded by the contractive affine transformation. Given the mean of each range block and the set of block transformations, the proposed

scheme finds for each image block the domain block and the transformation parameters that minimize the distortion between the image block and the transformed domain block. For N1 domain blocks (vectors of size k, k=BxB), N1 distortion computations are needed to determine the best match of an input range block. For a large number of domain blocks, the determination process is very time consuming. To keep almost the same distortion achieved by full search and to speed up the encoding process, partial domain block searches are simple and effective [9], [10], [15], [16]. The following method helps in identifying the domain blocks that are redundant and the same are eliminated in the search.

Let v_1, v_2 and v_3 be three orthogonal vectors, where

v_1 = ¼ [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] (3a)

v_2 = ¼ [1,1,1,1,1,1,1,1,-1,-1,-1,-1,-1,-1,-1,-1] (3b)

v_3 = ¼ [1,1,-1,-1, 1,1,-1,-1,1,1,-1,-1,1,1,-1,-1] (3c)

for k=16. The axis in the direction of v_i (i=1, 2, 3) is denoted as the ith axis. Let x_i be the projection value of an input block (vector) X on the i^{th} axis. That is, x_i is the inner product of X and v_i and can be calculated, as follows:

$$x_i = \langle X, v_i \rangle \quad (3d)$$

Similarly, denote c_{ki} as the projection value of a domain block W_k on the i^{th} axis. To speed the searching process, all domain block are sorted in ascending order of their projections on the first axis. Here x_1 is four times the mean value of X; x_2 and x_3 are the edge gradients in the vertical and horizontal directions, respectively, of X; and $[(x_2)^2 + (x_3)^2]$ represent the edge strength of X. Similar meanings are applied to c_{ki} (i=1,2,3). Let r be the distance between an input block (vector) X and a domain block W_j. If domain block W_j cannot satisfy the following condition, it will be rejected directly in the process of finding the closest domain block of X.

$$\left| c_{ji} - x_i \right| < r, \text{ i=1, 2, 3} \quad (4a)$$

where x_i and c_{ji} are the projection values of X and W_j, respectively, on the i^{th} axis. As shown in condition (4a), a smaller value of r will give a better performance of rejecting unlikely domain block. If the domain block W_i is the closest domain block of X, then their projection values on the first axis may be very close. As stated before, the projection value on the first axis of a vector is four times the mean value of the vector. Therefore, the domain blocks W_i, whose mean value is close to the mean value of X, is chosen as the initial domain blocks for that range block.

An additional condition to reduce the distortion computations is also used. To reject irrelevant domain block, the following condition is used accompany with condition (4a) to reject unlikely domain block in the process of finding the closest domain block of an input range block. Let c_j be the projection of the domain block W_j on the space spanned by v_1, v_2 and v_3, where

$$c_j = c_{j1}v_1 + c_{j2}v_2 + c_{j3}v_3 = \sum_{i=1}^{3} \langle W_j, v_i \rangle v_i \quad (4b)$$

Similarly, denote x as the projection of the input range vector X on the space spanned by v_1, v_2 and v_3, where, $x = x_1v_1 + x_2v_2 + x_3v_3 = \sum_{i=1}^{3} \langle X_j, v_i \rangle v_i$

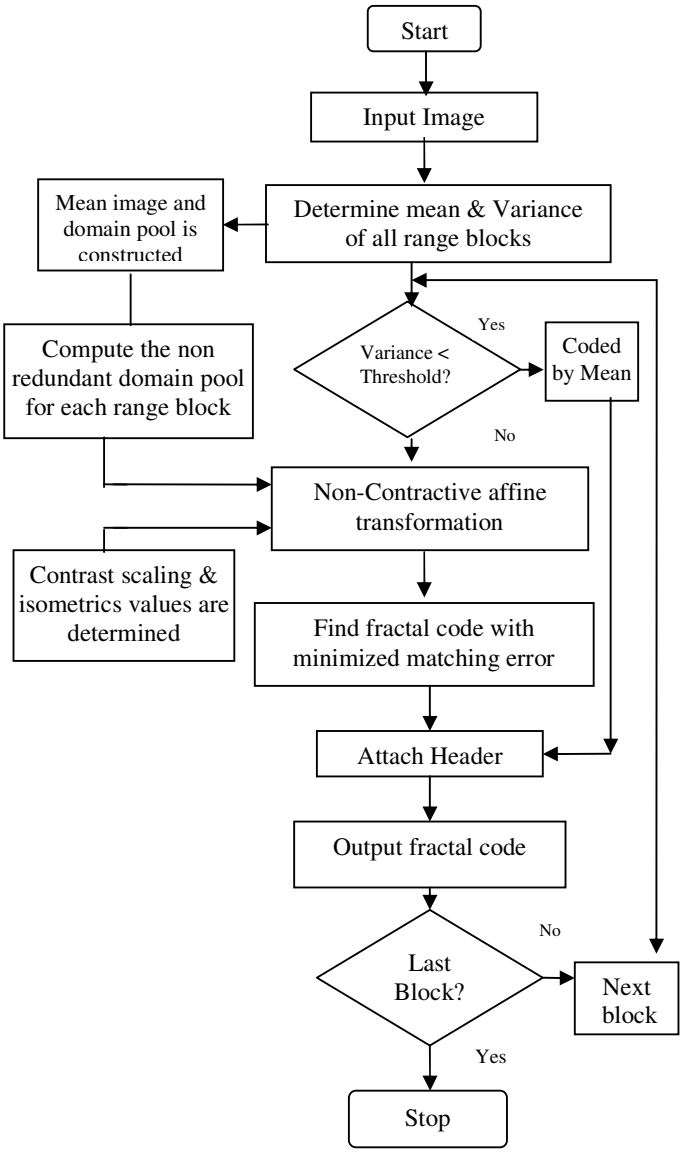


Fig.2. Flow chart of the encoder for the proposed method

Let $s_x = X - x$ and $s_{cj} = W_j - c_j$. From the definitions of c_j, x, s_{cj} and s_x, $c_j \perp s_{cj}, c_j \perp s_x, x \perp s_{cj}$ and $x \perp s_x$. A candidate domain block W_j should satisfy the following condition:

$$\left[(c_{j1}-x_1)^2+(c_{j2}-x_2)^2+(c_{j3}-x_3)^2\right]+(|s_{cj}|-|s_x|)^2<r^2 \quad (5)$$

That is, if the domain block W_j cannot satisfy condition (5), it will be discarded directly in the process of finding the closest domain block of X. Condition (5) activates only when the domain block cannot be rejected by using condition (4a). The texture vector (block) has a small value of $[(x_2)^2 + (x_3)^2]$ and a large value of $(|s_x|)^2$, which is called the texture strength of X; an edge block X posses a large value of $[(x_2)^2 + (x_3)^2]$ and a small value of $(|s_x|)^2$ and a smooth block X gives a small value of $[(x_2)^2 + (x_3)^2]$ and $(|s_x|)^2$. The x_1 is four times the mean value of X. The same characteristics are also applied to all domain block. A smooth domain block mainly uses its projection value on the first axis to distinguish itself from other smooth domain block;

an edge domain block distinguishes itself from other edge domain block using all three projection values; a texture domain block uses the texture strength and the projection value on the first axis to distinguish itself from other texture domain block. That is, condition (5) uses three features namely mean value, edge strength, and texture strength of a vector to reject unlikely domain block. Therefore, (5) has a good performance of rejecting unlikely domain block for an input range block if a good initial domain blocks is found. Another condition for terminating the searching process, if the distance r between a domain block W_i and input range block X is smaller than half the distance between W_i and any other domain block, then the domain block W_i must be the best match of the training vector X. Thus, the searching process may be stopped and W_i may be chosen as the closest domain block when it satisfies (6).

$$d(X,W_i) \leq 0.5 \min (d(W_j, W_i), j=1,2,...,i-1,i+1,\ldots N) \quad (6)$$

Let $d_{ni} = 0.5 \min (d(W_i, W_j), j=1,2,\ldots,i-1,i+1,\ldots N)$, where d_{ni} is half the minimum distance between W_i and all other domain block.

Thus using the above method the best domain block for each of the range block can be determined quickly. The new contractive affine transformation given in equ (1) can be expressed as,

$$\widehat{R} = i\{\alpha . D + \mu_R - \alpha . \mu_D\} = i\{\alpha . (D - \mu_D) + \mu_R\}$$

$$= i\{\ \alpha\ .\ (W_i - \text{mean}\ (W_i)) + \text{mean}(R)\} \quad (7)$$

The transformations applied to the minimized domain pool are luminance shift and isometries. The size of the domain blocks is the same as that of the range block and thus the contraction procedure in fractal coding schemes is eliminated. Therefore a new contractive affine transformation between the range block and the domain blocks in the minimized domain pool is calculated. The parameters used in the new contractive affine transformation are specified as follows. The index of the domain block in the domain pool is coded using $\log_2 N$ bits. The luminance shift is replaced by the mean which is coded using 8 bits. The contrast scaling α is determined by testing all the values in the following set {n/4, n=1, 2, 3, 4} to find the best one that minimizes the distortion [5] and is stored using 2 bits. On the other hand, the eight isometrics for shuffling the pixels in the block can be coded by three bits. An advantage of coding using this format is that it can be decoded to any size either enlarge or minimized depending on the requirements because isometry transformations are used. The distortion between the original and the coded range block is represented by the mean-squared-error (MSE) measurement defined as

$$\text{MSE}\ (R,\widehat{R}) = \frac{1}{B^2} \sum_{0 \leq i,i \leq B} \left(r_{i,j} - \widehat{r_{i,j}} \right)^2 \quad (8)$$

Generally, an image is coded block by block in a raster scan order, that is, from left to right and top to bottom. In other words, the fractal codes are generated in the raster scan order. In the proposed algorithm, each range block is also processed in the raster scan order. The fractal codes obtained can be stored efficiently by comparing it with the previous codes. First, the current processed range block is checked to see whether its adjacent left entry and its adjacent upper entry have the same value as itself [9]. If the same values for isometry, contract scaling and index of the domain pool are found in either of the two positions, only one bit will be used to indicate which one of the two entries has the same values as the current processed range block and the mean value alone is coded. Suppose that *x* bits are needed to represent the code in transmission, 9 or (x+1) bits are required for the current processed range block having the same value or not having the same value, respectively. Note that an extra indicator bit is needed to distinguish whether the same value of the current block has been found or not. If both entries do not have the same values for the above mentioned parameters as the current one, the current value is then checked to see whether the relative addressing technique can be employed or not. As the domain blocks used are sorted prior by the mean values of the domain block, the resultant values tends to be more compact. In other words, the neighboring codes are quite similar to each other and the differences between any two of them are small. The relative addressing technique is employed to improve the compression performance. For each value that does not have the same value as its adjacent left code and its adjacent upper code, the offset between this range block and its previously processed range block is computed. If the offset values are smaller than the predefined threshold, the relative offset values are transmitted to the decoder. Otherwise, the original values for this range block are transmitted to the decoder. An extra bit is needed in transmission to indicate the two different types.

3.2 DECODER

Fig. 3 shows the flow chart of the decoder in the proposed enhanced iteration-free fractal scheme. The entire fractal codes are first received and determined whether or not the range block is coded by the mean from its header. The mean image is reconstructed with the mean information in the codes. This mean image is identical to the mean image used in the encoder since both are constructed by the same block means. Therefore, the domain blocks generated from both the mean image will be the same. If the block is coded by the mean, the value of each pixel in the decoded block is equal to the mean value. Otherwise the contractive affine transformation is performed to reconstruct the coded range block. The decoding process ends when the last range block is reconstructed. Only the fixed mean image that is reconstructed from the received codes is required for the construction of the domain pool. On the other hand, the range blocks can be decoded in parallel. Therefore, the proposed decoder is very much suitable for the hardware implementation and high-speed applications.

Using smaller domain pools, the number of accesses to the domain pool memory and the power consumed per memory access are reduced [14]. The use of smaller domain pools also leads to reduction of the number of executions of the distortion criterion since smaller numbers of candidate domain pool exist. So reduction in search for the best domain block was achieved by using the partial domain block search. This also leads to significant power savings since the computation of the distortion criterion forms a significant part of the total coding computation, and computational reduction is equivalent to power consumption reduction.

The algorithm of the proposed enhanced iteration free fractal image coding is given as follows:

Encoder:

Step1: Partition the given image into range blocks X of size BxB and find the mean and variance of each X.

Step2: Plot the mean image using the mean of X as the pixel value and partition this mean image into blocks of size BxB and apply LBG algorithm to get the domain pool W of the required size N (N=16 / 32 / 64).

Step3: Determine the domain pool's projection value on the first axis and arrange them in the ascending order of the projection values on the first axis. Determine the projection value c_{ij} and $|s_{ci}|$ (i=1, 2… N and j= 1, 2, 3) for all domain blocks in the domain pool. Construct the nearest distance table dt={d_{n1}, d_{n2}… d_{nN}}.

For each range block X:

Step4: If variance (X) < E assign 0 to label and μ_x to code. Process the next range block.

Step5: Assign 1 to label. Choose the domain block W_m and compute the distance between X and W_m, where W_m satisfies the following condition $|x_1 - c_{m1}| \leq |x_1 - c_{j1}|$, $1 \leq j \leq N$, and $j \neq m$. Let $r = d(X, W_m)$ and store the value of r^2 to sqr. If $r \leq d_{nm}$, then W_m is the closest domain block of X. Go to step 11. Otherwise compute the projection values x_i (i=1, 2, 3) and $|s_x|$ of X. Set d=1.

Step6: If (m+d) $\geq$ N or the domain block W_{m+d} is deleted, go to step 8. Otherwise go to step 7.

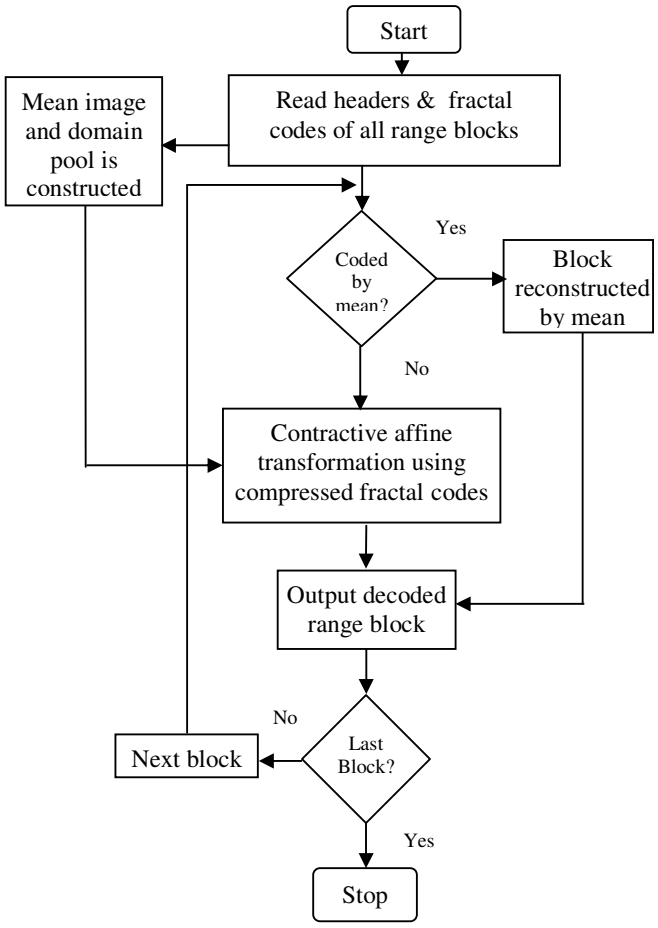


Fig.3. Flow chart of the decoder for the proposed method

Step7: a) Compute $D_i = | c_{(m+d)i} - x_i |$ (i=1,2,3).

If $D_1 \geq r$, then eliminate all domain blocks from W_{m+d} to W_n and go to step 8.

If $D_j \geq r$, (j=2,3) then delete domain blocks W_{m+d} and go to step 8.

b) Compute $D_t = \sum_{i=1}^{3} D_i^2 + \left(|S_{c(m+d)}| - |S_x|\right)^2$. If $D_t \geq$ sqr, then delete domain blocks W_{m+d} and go to step 8.

c) Compute r' = d (X, W_{m+d}) and set sqr' = $(r')^2$. If r' $\geq$ r then domain block W_{m+d} is eliminated and go to step 8. Otherwise set r = r' and sqr = sqr'. If $r \leq d_{n(m+d)}$, then W_{m+d} is the closest domain block of X, go to step 11. Otherwise go to step 8.

Step8: If (m-d) < 0 or the domain block W is deleted, go to step 10. Otherwise go to step 9

Step9: a) Compute $D_i = | c_{(m-d)i} - x_i |$. i=1, 2, 3.

If $D_1 \geq r$, then eliminate all domain blocks from W_{m-d} to W_0

If $D_j \geq r$, (j=2,3) then delete domain blocks W_{m-d} . Go to step 10.

b) Compute $D_t = \sum_{i=1}^{3} D_i^2 + \left(|S_{c(m+d)}| - |S_x|\right)^2$. If $D_t \geq$ sqr, then delete domain blocks Wm_{-d} and go to step 10.

c) Compute r' = d(X,W_{m-d}) and set sqr' = $(r')^2$. If r' $\geq$ r then delete domain block W_{m-d} , go to step 10. Otherwise set r = r' and sqr = sqr'. If r $\leq d_{n(m-d)}$, then W_{m-d} is the closest domain block of X, go to step 11 otherwise go to step 10.

Step10: Set d = d+1. If (m+d > N and m-d < 0) or (both W_{m+d} and W_{m-d} are deleted), go to step 11. Otherwise, go to step 6.

Step11: Apply the isometry transformations *i* to the minimized domain pool W for contrast scaling $\alpha = x/4$ {for x=1 to 4}. Calculate the RMS error between the transformed domain blocks and the range block. Transfer the values of *i*,α , μ_x, index of W_j, which has the minimum RMS error to code. Process the next range block

Decoder:

Step1: Read the header and the minimized fractal codes

Step2: Extract the mean information of each range block from the labels and compute the domain pool.

Step3: For each range block check if coded by mean value construct the block using the mean. Otherwise extract the values for i, α and index of W_j, apply the transformation to W_j and construct the block using the transformed values of W_j. Process the next range block.

The performance of the quality of the decoded image of size MxN with 8 bit gray scale resolution is measured using peak signal-to-noise-ratio (PSNR) given in Eq.(9) and the bit rate (the required bits per pixel) using Eq.(10)

$$PSNR(dB) = 20\log_{10}\frac{(2^8-1)}{\left[\left(\sum_{1\le i\le M}\sum_{1\le j\le N}(r_{i,j}-\hat{r}_{i,j})^2\right)\Big/(MxN)\right]^{1/2}} \quad (9)$$

$$\text{Bit rate} = \frac{N_\mu * I_\mu + N_f * I_f}{(MxN)^2} \text{ bit/pixel} \quad (10)$$

where

N_μ number of blocks coded by mean

N_f number of blocks coded by fractal codes

I_μ required bits for the block mean

N_f required bits for (block mean + isometry + contrast scaling + domain block number).

4. IMPLEMENTATION OF ENHANCED ITERATION FREE FRACTAL CODING

In computer simulation, four 512 x 512 benchmark images Lena, Taj Mahal, Jet Plane and Building [shown in Fig. 4(a) - (d)] with eight-bit grayscale resolution are used to test the proposed enhanced iteration-free fractal coding scheme. In the simulation, the images were partitioned into range blocks with the single size, either 8x8 or 4x4 or 2x2 or with two-level quad tree partition of sizes (8x8 and 4x4) or (4x4 and 2x2). The maximum block size is set to 8x8 because for a range block size greater than 8x8 the determination of the proper domain block was difficult and the quality of the image reconstructed was poor. The threshold value E for the variance of range blocks was chosen by trial and error basis to be of size 20 for block size 8x8, 10 for 4x4 and 5 for 2x2 that results in good compression ratio and PSNR. The number of blocks in the mean image is the size of the domain pool. Domain pool design of 3 sizes was used. N_D = 16, 32 and 64.

(a) Lena (b) Taj Mahal

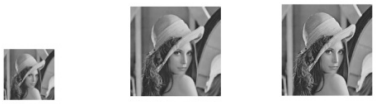
(c) Jet Plane (d) Building

Fig.4. Original (512 x 512, 8 bit/pixel) images

4.1 SINGLE BLOCK SIZE

The range block with a size (8x8, 4x4 and 2x2) was considered for simulation. The length of the attached header to the existing iteration free fractal code for each range block was only one bit because it only denotes whether or not the range block is coded by the mean. In this proposed method the number of headers are two – one header of size one bit to denotes whether or not the range block is coded by the mean and the other of size two bit to denote whether the range blocks code is the same as the adjacent range block or by the adjacent top block or by a relative displacement of less than 4 or by the original code. For an image partitioned by 4x4 range blocks, every block mean was calculated and a 128x128 mean image was obtained. Fig. 5(b) shows that the mean image of Lena got by this partition and it is very similar to its original image except its size. Therefore the domain pools of different sizes namely 16, 32 and 64 using the LBG-based method from the mean image was constructed.

The coding performance with the contractive affine transformation under the different sizes for the domain pool on the parameters like coding time, image quality and bit rate was determined. For the image partitioned by 8x8 and 2x2 range blocks, the 64x64 and 256x256 mean image for Lena was obtained and shown in Fig. 5(a) and 5(c) respectively. The domain pools of different sizes were constructed on these images using the LBG-based method and computed the coding performance on the same parameters for different sizes of the domain pool.



(a) 64x64 (b) 128x128 (c) 256x256

Fig.5. Mean images of Lena

4.2 TWO-LEVEL BLOCK SIZES

From the results shown in Table 1, the chosen block size greatly affects the encoding time, memory size and the MSE of the coded image. In order to compromise the memory size and coding time, partitioning the image into the range blocks with two-level (parent 8x8 and child 4x4) sizes was performed [11]-[13],[17]. An image is first partitioned into parent range blocks and the coding procedures are the same as that in Section 3.1. If the parent range block is coded by the contractive affine transformation and the distortion between the original and coded range blocks, MSE ($R_8, \hat{R}_8$), is greater than the threshold value E=10, the parent range block is split into four child range blocks. The coding procedures for the child range block are the same as that described in Section 3.1 but the threshold for the variance is taken to be as E/2. Now, the bit rate is affected by the number of the partitioned parent and child range blocks. The more the parent range blocks in the coded image, the lower the final bit rate. In order to verify that the proposed method also perform well for other images, the simulation results for the Taj Mahal, Jet Plane and Building [shown in Fig. 4(b) – 4(d)] are also given.

5. RESULTS AND DISCUSSION

Experimental results of the coding time, memory requirements and image quality using different sizes of codebooks for the iteration-free fractal codes and the proposed enhanced iteration free fractal method are tabulated in Table 1. The coding time of the simulation results for the Lena image for a block size of 8x8 is shown in Fig. 6(a). From the graph it is observed that the encoding time for the proposed method is almost the same for domain pool of different sizes, but the encoding time nearly doubles itself when the domain pool increases using the iteration free fractal coding method. The coding time of the simulation results based on the 4x4 and 2x2 sizes for the domain pool are shown in Fig. 6(b) and 6(c). The encoding time for 2 x 2 block size for Lena image using the proposed method is greater when compared to the iteration free fractal method because all the input blocks are coded by the mean value in both the proposed and iteration free fractal method but the overheads the proposed method has to perform takes a little extra time. This is clearly seen when the same block size is used for the images Taj Mahal, Jet Plane and Building where the blocks are coded by the mean value and affine transformations. The proposed method provides better performances than the iteration free fractal coding scheme in terms of coding time and storage capacity. As the size of the mean image increases, the quality of the image becomes more nearer to the iteration free fractal method. The MSE of the decoded image partitioned by the 8x8 block size is higher than that partitioned by the 4x4 and 2x2 block size since a smaller block size leads to a smaller matching error for the affine transformation. However, the bit rate increases significantly because the number of the 4x4 and 2x2 range blocks are four times and sixteen times the number of the 8x8 range blocks. The decompressed image of Lena, Taj Mahal, Jet Plane and Building for the single block partition of sizes 8x8, 4x4 and 2x2 using the proposed method are shown in Fig 8, 11,13 and 15 respectively. The decompressed image of Lena for the single block partition of sizes 8x8, 4x4 and 2x2 using the existing iteration free fractal method is shown in Fig 7.

Quad tree partitioning was performed to reduce the coding time and memory size. Experimental results of the coding time, memory capacity and image quality using different sizes of codebooks with 2 level quad tree partitioning of sizes 8x8 & 4x4 and 4x4 & 2x2 are tabulated in Table 2. The coding time of the simulation results of the Lena image based on the proposed enhanced iteration-free fractal coding and iteration-free fractal coding schemes using quad tree partitioning is shown in Fig. 9. The decompressed image of Lena, Taj Mahal, Jet Plane and Building for the two level block partition of sizes 8x8&4x4 and 4x4&2x2 are shown in Fig 10, 12, 14, 16 respectively. As in the single block partitioning the coding time of the proposed method is almost the same as the domain pool size increases but in the iteration free fractal method the time nearly doubles itself as the domain pool doubles. Using two-level block sizes, the resultant coding time and memory size of the proposed methods are within a moderate range. The compression ratio using the proposed method is 37.76, 38.89, 37.86 and 30.65 respectively for the images Lena, Taj Mahal, Jet Plane and Building using the single block size of 8 x 8 and it is 20.7, 17.2, 18.7 and 10.7 respectively for the images Lena, Taj Mahal, Jet Plane and Building using the quad tree partition for the block size of 8 x 8 and 4 x 4.

Table.1. Experimental results of the, coding time, memory requirements and image quality using different sizes of codebooks

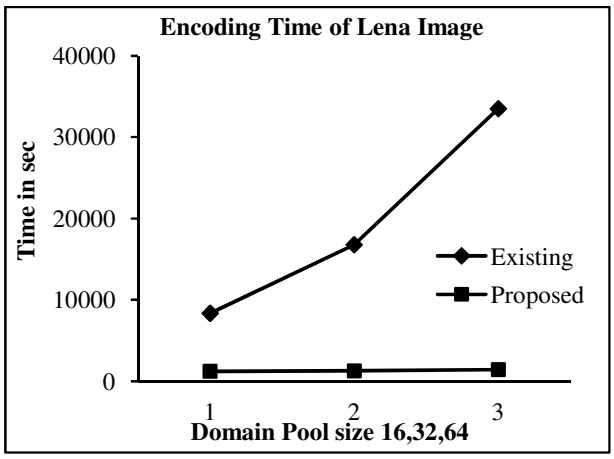
Image	Type of Encoding	Range Block Size		16 level		32 level		64 level		
				RMS	Coding Time (secs)	RMS	Coding Time (secs)	PSNR	RMS	Coding Time (secs)
Lena	Single Level	8x8	Existing	9.4234	8346	9.5166	16794	28.5	9.5947	33486
			Proposed	12.1484	1206	11.1746	1289	27.3	10.9575	1442
		4x4	Existing	5.9490	43081	6.0160	89128	32.4	6.0946	142480
			Proposed	6.6930	3738	6.5831	4154	31.7	6.6279	4955
		2x2	Existing	0.7200	211	0.7200	210	51	0.7200	214
			Proposed	0.7181	360.296	0.7181	374.359	51	0.7181	348.236
Taj Mahal	Single Level	8x8	Existing	10.9459	7715	11.0452	15547	27.3	11.0562	31256
			Proposed	16.3411	1158	14.8646	1199	25.3	13.7929	1300
		4x4	Existing	7.5839	47413	7.5420	95429	30.4	7.6241	190728
			Proposed	8.9749	5109	8.4748	5852	30.0	8.1079	6722
		2x2	Existing	6.38803	120830	6.3002	249680	32.1	6.2874	491476
			Proposed	6.3764	14341	6.0505	19830	32.7	5.8807	29000
Jet Plane	Single Level	8x8	Existing	10.1130	7538	10.1288	15189	27.9	10.2217	30257
			Proposed	13.5390	1086	12.7988	1201	26.4	12.2329	1425
		4x4	Existing	6.9741	40075	6.9004	81143	31.3	6.8954	164005
			Proposed	7.4735	3562	7.3867	4265	30.7	7.4325	5464
		2x2	Existing	4.0101	53677	3.9904	112367	36.1	3.9877	214790
			Proposed	4.2113	7158	4.1626	11435	35.8	4.1414	16310

Building	Single Level	8x8	Existing	15.8962	13225	15.9321	26954	24.1	15.9942	54167
			Proposed	18.2354	1957	17.7019	2124	23.4	17.2052	2448
		4x4	Existing	11.4789	56522	11.5123	114589	26.9	11.4467	230038
			Proposed	12.6340	6467	12.4717	7745	26.2	12.5380	10274
		2x2	Existing	6.4957	178240	6.4623	357594	31.9	6.4333	717856
			Proposed	7.0891	23712	7.0378	33517	31.3	6.9271	49517

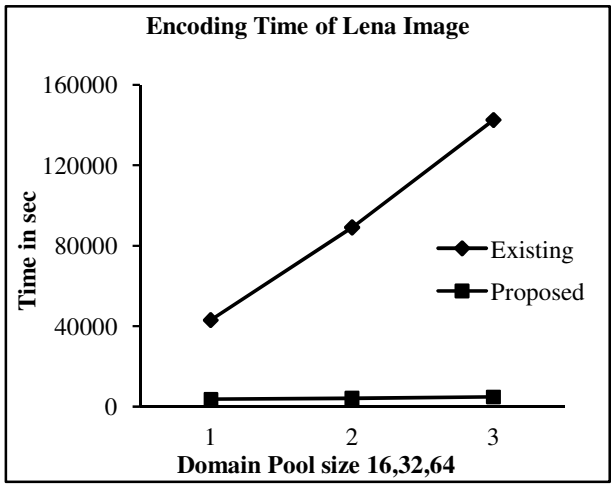
Table.2. Experimental results of the coding time, memory capacity and image quality using different sizes of codebooks with 2 level quad tree partitioning

Image	Type of Encoding	Range Block Size		16 level		32 level		64 level			
				RMS	Coding Time (secs)	RMS	Coding Time (secs)	PSNR	RMS	Coding Time (secs)	Memory Size
Lena	Two Level	8x8, 4x4	Existing	5.7923	38478	5.6865	78858	32.9	5.7429	157942	21.44
			Proposed	6.2847	4633	6.3228	5288	31.9	6.4878	6403	11.24
		4x4, 2x2	Existing	1.7605	35295	1.7613	71633	43.2	1.7638	143320	28.05
			Proposed	1.7443	4188	1.7393	4732	43.3	1.7389	5732	24.16
Taj Mahal	Two Level	8x8, 4x4	Existing	6.7785	39378	6.8941	79756	31.3	6.9124	169634	16.20
			Proposed	7.9569	5155	7.9843	5651	29.9	8.1285	6587	13.85
		4x4, 2x2	Existing	5.1595	96360	5.1699	98749	33.9	5.1502	394576	56.74
			Proposed	5.1519	14494	5.1726	18638	34.1	5.0272	26421	37.66
Jet Plane	Two Level	8x8, 4x4	Existing	6.4830	35775	6.4256	72458	32.0	6.3879	145764	17.36
			Proposed	7.2673	4529	7.2628	5389	30.8	7.3267	7107	12.82
		4x4, 2x2	Existing	4.2083	64343	4.1567	135745	36.0	4.0126	279432	52.74
			Proposed	4.2541	10893	4.0863	14027	36.0	4.0517	20833	30.72
Building	Two Level	8x8, 4x4	Existing	11.6764	65500	11.6189	132789	26.8	11.5786	274897	22.74
			Proposed	12.1794	7822	12.0826	9551	26.4	12.2714	12026	20.25
		4x4, 2x2	Existing	6.7237	190856	6.6999	389456	31.6	6.6834	784267	80.41
			Proposed	6.5455	29285	6.3301	38468	32.1	6.3318	58100	56.18

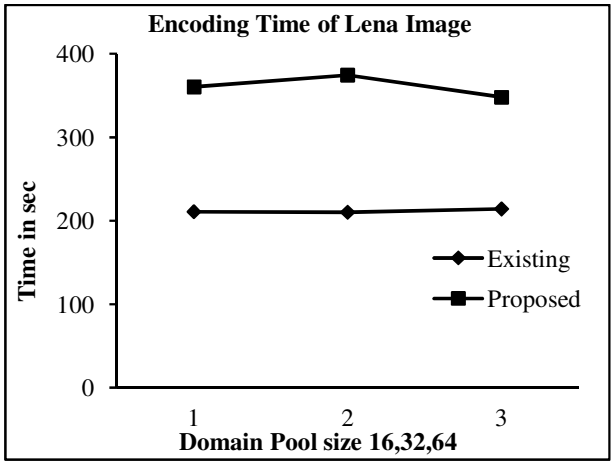
Apparently, the performances of this method based on coding time and storage requirements are greatly improved by using the proposed method.



(a) Block Size 8 x 8



(b) Block Size 4 x 4



(c) Block size 2x2

Fig.6. Coding Time of Lena using block size 8x8, 4x4 and 2x2

(a) Block size 8x8

(b) Block size 4x4

(c) Block size 2x2

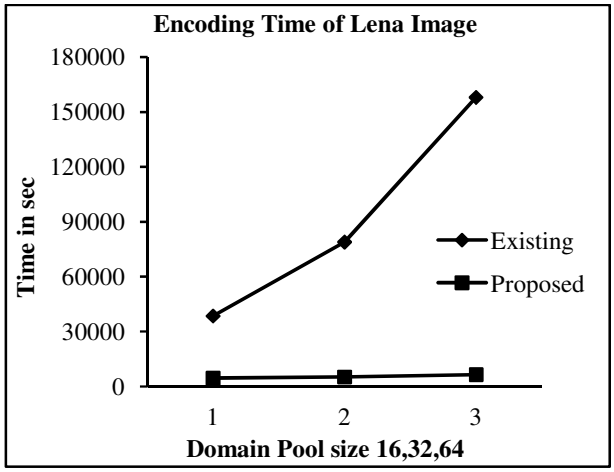
Fig.7. Decompressed Image of Lena for Block Size 8x8, 4x4 and 2x2 Using The Iteration free Fractal Coding

(a) Block size 8x8

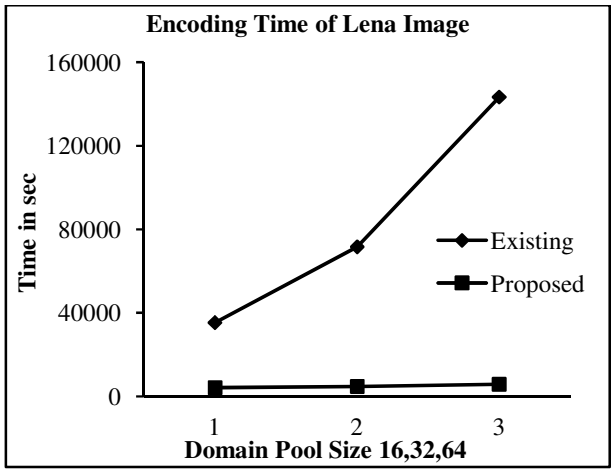
(b) Block size 4x4

(c) Block size 2x2

Fig.8. Decompressed Image of Lena for Block Size 8x8, 4x4 and 2x2 Using The Proposed VQ Technique



(a) Block Size 8x8 & 4x4



(b) Block size 4x4 & 2x2

Fig.9. Coding time of Lena using 2 level Quad tree partitioning of block size 8x8, 4x4 and 4x4, 2x2

(a) Block size 8x8 & 4x4

(b) Block size 4x4 & 2x2

Fig.10. Decompressed Image of Lena for Two-Level Block Size 8x8 & 4x4 and 4x4 & 2x2 Using The Proposed VQ Technique

(a) Block size 8x8

(b) Block size 4x4

(c) Block size 2x2

Fig.11. Decompressed Image of Taj Mahal for Block Size 8x8, 4x4 and 2x2 Using The Proposed VQ Technique

(a) Block size 8x8 & 4x4

(b) Block size 4x4&2x2

Fig.12. Decompressed Image of Taj Mahal for Block Size 8x8 & 4x4 and 4x4 & 2x2 Using The Proposed VQ Technique

(a) Block size 8x8

(b) Block size 4x4

(c) Block size 2x2

Fig.13. Decompressed Image of Jet Plane for Block Size 8x8, 4x4 and 2x2 Using The Proposed VQ Technique

(a) Block size 8x8 & 4x4

(b) Block size 4x4 & 2x2

Fig.14. Decompressed Image of Jet Plane for Block Size 8x8 & 4x4 and 4x4 & 2x2 Using The Proposed VQ Technique

(a) Block size 8x8

(b) Block size 4x4

(c) Block size 2x2

Fig.15. Decompressed image of Building for block size 8x8, 4x4 and 2x2 using the proposed VQ technique

(a) Block size 8x8 & 4x4

(b) Block size 4x4 & 2x2

Fig.16. Decompressed Image of Building for Block Size 8x8 & 4x4 and 4x4 & 2x2 Using The Proposed VQ Technique

6. MERITS AND APPLICATIONS OF THE PROPOSED SCHEME

Image and video coding form an integral part of information exchange. They are not confined only to immobile environment. They are also used in mobile and wireless communications. Rapid development of the Internet with its new services and applications has created fresh challenges for the further development of mobile communication systems. A simple example is that mobile phones (among many other new media-centric devices) now have high resolution cameras attached, and are also capable of displaying video. The growing demand for mobile services has led to a worldwide reconsideration of established methods of transmitting images in a compressed and efficient form with minimum time. The proposed method has the advantages such as low time consumption and less memory requirements for storage which is most needed in today's communication.

7. CONCLUSION

In this paper, a new, fast-encoding algorithm for iteration free fractal image coding is introduced. This algorithm uses three features namely mean value, edge strength, and texture strength of a vector to eliminate many of the unlikely domain blocks from the domain pool, which is not available in the existing algorithm. The proposed algorithm has the best

performance in terms of computing time and storage space. Compared with iteration free fractal code method, the proposed enhanced iteration free fractal algorithm can reduce the number of distortion calculations there by reducing the coding time to more than 5.7 times and 11.5 times for the domain block of size 8x8 and 4x4 for the Lena image for the domain pool of size 16. This time is further reduced as the size of the domain pool increases. It further reduces the memory requirements for storing the fractal codes than the iteration free fractal image code.

REFERENCES

[1] E. Jacquin, 1992, "Image coding based on a fractal theory of iterated contractive image transformations". IEEE Trans. Image Processing, Vol. 1, No.1, pp. 18–30.

[2] A. E. Jacquin, 1993, "Fractal image coding: A review". Proc. IEEE, Vol. 81, No.10, pp. 1451–1465.

[3] R. Hamzaoui, 1996, "Decoding algorithm for fractal image compression". Electronics Letters, Vol. 32, No.14, pp. 1273-1274.

[4] G. Lu and T.L. Yew, 1994, "Image compression using quadtree partitioned iterated function systems". Electronics Letters, Vol. 30, No.1, pp. 23-24.

[5] B. Wohlberg and G. de Jager, 1999, "A review of the fractal image coding literature" lEEE Trans Image Processing, Vol 8, No.12, pp 1716-1729.

[6] Y. Linde, A. Buzo, and R. Gray, 1980, "An algorithm for vector quantization design". IEEE Trans. Commun., Vol.28, No.1, pp. 84–95.

[7] N.M. Nasrabadi and R.A. King, 1988, "Image coding using vector quantization: a review". IEEE Trans.Com., Vo1.36, No.8, pp. 957-971.

[8] Hsuan T. Chang and Chung J. Kuo, 2000, "Iteration-Free Fractal Image Coding Based on Efficient Domain Pool Design". IEEE Trans. Image Processing, Vol. 9, No 3, pp. 329-339.

[9] Jim Z. C. Lai and Yi-Ching Liaw, 2004, "Fast-Searching Algorithm for Vector Quantization Using Projection and Triangular Inequality". IEEE Transactions on Image Processing, Vol. 13, No. 12, pp 1554-1558.

[10] Yu-Chen Hu and Chin-Chen Chang, 1999, "Low Complexity Index-Compressed Vector Quantization For Image Compression". IEEE Transactions on Consumer Electronics, Vol. 45, No. 1, pp 219-224.

[11] H. Yuen and T. P. J. To, 1996, "Efficient Variable Rate Vector Quantization Using Quad tree Segmentation". IEEE Transactions on Consumer Electronics, Vol. 42, No. 2, pp. 212-215.

[12] Guy Cazuguel, Andras Cziho, Base1 Solaiman, Christian Roux, Michel Robaszkiewicz, 1997, "Improving spatial vector quantization for image compression by use of a quadtree scheme. Application to echo endoscopic image compression". Annual International Conference of the Engineering in Medicine and Biology Society, pp.894–897.

[13] Yu-Chen Hu and Chin-Chen Chang, 1999, "Variable Rate Vector Quantization Scheme Based on Quadtree Segmentation". lEEE Transactions 011 Consumer Electronics, Vol. 45, No. 2, pp.310 – 317.

[14] K. Masselos, P. Merakos, T. Stouraitis, and C. E. Goutis, 1999, **"**Novel Vector Quantization Based Algorithms for Low-Power Image Coding and Decoding". IEEE Tran. Circuits and Systems—II: Analog And Digital Signal Processing, Vol. 46, No. 2, pp. 193-198.

[15] Yung-Gi, Wu, Ming-Zhi, Huang, Yu-Ling, Wen, 2003, "Fractal Image Compression with Variance and Mean". ICME, Vol.1, pp. 353 – 356.

[16] K. Masselos, T. Stouraitis, C. E. Goutis, 1998, "Novel CodeBook Generation Algorithms for Vector Quantization Image Compression". IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 5, pp, 2661-2664.

[17] D. Saupe and S. Jacob, 1997, "Variance-based quadtrees in Fractal image Compression". Electronics Letters, Vol.33, No.1, pp 46-48.