

# An Adaptive Fractal Image Compression

Taha mohammed Hasan<sup>1,2</sup> and Xingqian Wu<sup>1</sup>

<sup>1</sup> School of Computer Science and Technology, Harbin Institute of Technology  
Harbin, 150001, China

<sup>2</sup> College of Science, University of Diyala, Iraq

## Abstract

In this paper an Adaptive Fractal Image Compression (AFIC) algorithm is proposed to reduce the long time of the Fractal Image Compression (FIC). AFIC worked on; minimizing the complexity and the number of matching operations by reducing both of the range and domain blocks needed in the matching process, for this purpose Zero Mean Intensity Level Fractal Image Compression based on Quadtree partitioning, Variance Factor Range Exclusion, Variance Factor Domain Selection and Domain Pool Reduction techniques is used. This in turn will affect the encoding time, compression ratio and the image quality. The results show that AFIC significantly speeds up the encoding process and achieves a higher compression ratio, with a slight diminution in the quality of the reconstructed image. In comparison with some resent methods, the proposed method spends much less encoding time, get higher compression ratio while the quality of the reconstructed images is almost the same.

**Keywords:** *Fractal, range block, quadtree, variance, image compression, encoding time*

## 1. Introduction

Compression and decompression technology of digital image has become an important aspect in the storing and transferring of digital image in information society [1]. At present, fractal image compression has become one of the most promising encoding technology in the new generation of image compression for its novel idea, high compression ratio and resolution independence [2], [3]. The basic idea of fractal image compression technique was introduced by Barnsley et al in 1988. [4], [5]. The underlying idea was inspired from the fact that all (or almost all) of natural environment photographs generally shows self-similarity on different scales [6]. Therefore considerable amount of redundancy are implied in the images by this self-similarity property.

By the means of the Iterated Function System proposed by Barnsley, there is a contractive transformation for each image that has the fixed-point identical to the image itself. In other words, applying that transform (function) iteratively on an arbitrary starting image, the result converges to the original image. Thus, the image is encoded by the transformation [7].

The practical coding algorithm was not realized until 1992 by Jacquin [8]. His algorithm is based on the Partitioned Iteration Function System (PIFS) which can achieve a high compression ratio and good retrieved image quality by utilizing the self-similarity characteristic that founded in different parts of the images [9], [10].

One of the most important characteristics of fractal image coding is its unsymmetrical property of encoding and decoding processing [11]. At encoding process, FIC method must do a large amount of similarity computations in order to find the best-matched domain block, so it is time-consuming [12]. While decoding algorithm is relatively simple and fast. Therefore, improving the encoding speed is an interesting research topic for FIC [13]. Many encoding techniques were presented by the researchers to speed-up the fractal encoder. These techniques include classification techniques [7], [14]-[16] quad-tree technique [18]-[20], spatial correlation [21], [22], and evolutionary computation technique [13], [23], [24]. In this paper an adaptive method is proposed to; reduce the long time of the FIC, increase the compression ratio and keep the reconstructed image quality. This method worked on: 1-Reducing the complexity of matching operations by using Zero Mean Intensity Level Fractal Image Compression (ZMIL FIC) which can speed up the encoding operation and increase both of the compression ratio and the reconstructed image quality as it illustrated in our previous work [25]. 2-Minmizing the number of

matching operations by reducing both of the range and domain blocks needed in the matching operations, for this purpose, an adaptive quadtree partitioning technique is used then three techniques have been used; the first one is called Range Exclusion (RE), which used a variance factor to reduce the number of range blocks by excluding the homogenous ranges from the matching process; the second one is called Variance Domain Selection (VDS), which searches only the domain blocks with small variance difference to the encoded range; the third one is called Reducing the Domain Image Size (RDIZ), it reduces the domain pool by minimizing the Domain Image Size to only  $1/16^{\text{th}}$  original image size. This in turn will affect the encoding time, compression ratio and the image quality. All these techniques are worked together under one method called Adaptive Fractal Image Compression (AFIC). The results show that AFIC significantly speeds up the encoding process and achieves a higher compression ratio, with a slight diminution in the quality of the reconstructed image. At about the same PSNR, the experimental results show that, the proposed method is about 21.68, 8.13 and 8.49 times faster than Duh's classification method [16], PSO-KI method [13] and Lin's EP-NRS method [24] respectively. Moreover it gets 1.324 more compression ratio, while the penalty of retrieved image quality only a decay of 0.66 dB.

The organization of this paper is as follows. In section 2 a self-similarity briefly explained. Section 3 describes the basic fractal image compression with the full search case. In section 4, our proposed method and its techniques are illustrated. Experimental results including encoding time, compression ratio, and peak signal-to-noise ratio (PSNR) are given in section 5. Section 6 presents the conclusions.

## 2. Self-Similarity

In mathematics, a self-similar object is exactly or approximately similar to a part of itself (i.e. the whole has the same shape as one or more of the parts). Many objects in the real world, such as coastlines, are statistically self-similar: parts of them show the same statistical properties at many scales. Self-similarity is a typical property of fractals. Scale invariance is an exact form of self-similarity where at any magnification there is a smaller piece of the object that is similar to the whole. For instance, a side of the Koch snowflake is both symmetrical and scale-invariant; it can be continually magnified 3x without changing shape [26].

Natural images are not exactly self-similar, natural images can be partially constructed from affine transformations of small parts of themselves. Self-Similarity indicates that small portions of the image resemble larger portions of the same image. The search for this resemblance forms the basis of fractal compression scheme [27]. Therefore the

image must be partitioned into blocks to find self-similarity in other portion of the same image. This is intrinsic of fractal encoding techniques.

Figure 1 shows some of the self-similar portions in Lena image, there is a reflection of the hat in the mirror. The reflected portion can be obtained using an affine transformation of a small portion of her hat. Parts of her shoulder are almost identical [28].



Fig. 1 An example shows the self-similarity in Lena image.

## 3. Basic Fractal Image Compression

The basic fractal image compression scheme (BFIC) is based on contractive transformations and PIFS in a two-dimensional metric space [29]. The PIFS, which is essentially a set of contraction mappings, are determined by analyzing the image. These mappings can exploit the self-similarity that is commonly present in most images. That is part **A** of a certain image is similar to another part **B** of the image, by doing an arbitrary number of contractive transformations that can bring **A** and **B** together. These contractive transformations are actually common geometrical operation such as rotation, scaling, skewing and shifting. By applying the resulting PIFS on an initially blank image iteratively can be completely reconstructed an approximation to the original image at the decoder [30]. BFIC encoder consists of the following stages:

- Create the range pool ( $R$ ) by partitioning the image of size  $m \times m$  into non-overlapped ranges blocks of size  $n \times n$ . So the number of range blocks  $r$  in the range pool  $R$  will be  $(m/n) \times (m/n)$ .
- Create domains blocks ( $D$ ). It will be all the possible overlapped blocks of size  $2n \times 2n$ . So the number of the domain blocks  $d$  in the domain pool  $D$  will be  $[(m-2n)/j+1] \times [(m-2n)/j+1]$ . Where  $j$  is the horizontal and vertical jump step across the domain pool. In order to execute the similarity measure between range block and domain block, the size of the domain block must be first sub-sampled to be  $n \times n$  such that its size is the same as  $r$ .

- Search for a best matched  $d_i$  block in  $D$  for each  $r$  block in  $R$  and find an affine transformation  $w_i$  that adjusts the intensity values in the  $d_i$  to those in the  $r$ . [31].

$$w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \quad (1)$$

So that when  $w_i$  applied to the  $d_i$  should get something that is very close to  $r_j$ . The crux, then of the encoding image is to find contractive maps  $w_i$  that minimize the distances between  $r_i$  and corresponding  $d_i$  see figure 2 [32].



Fig. 2 The transform between domain block ( $D_i$ ) and range block ( $R_j$ ).

This shows why fractal compression is a slow technique, since each range block must be compared to all domain blocks including eighth symmetry orientations, see table 1 [33]. This operation allows the best match to be found, the best matching between domain and range blocks which is satisfy the minimum distortion error  $E(r, d)$  of Eq. (2) [28].

$$E(r, d) = \frac{1}{n} \sum_{i=1}^n (s_i d_i + o - r_i)^2 \quad (2)$$

In other words, we seek to minimize the quality of distortion over  $d \in D$  in Eq. (3) with respect to the parameters scale ( $s$ ) and offset ( $o$ ) in Eq. (4) and (5) respectively [34].

$$E(R, D) = \frac{1}{n} \left[ \sum_{i=1}^n r_i^2 + s \left( s \sum_{i=1}^n d_i^2 - 2 \sum_{i=1}^n d_i r_i + 2o \sum_{i=1}^n d_i \right) + o \left( nO - 2 \sum_{i=1}^n r_i \right) \right] \quad (3)$$

Where:

$$s = \frac{n \left( \sum_{i=1}^n d_i r_i \right) - \left( \sum_{i=1}^n d_i \right) \left( \sum_{i=1}^n r_i \right)}{\left[ n \sum_{i=1}^n d_i^2 - \left( \sum_{i=1}^n d_i \right)^2 \right]} \quad (4)$$

And

$$o = \frac{\left[ \sum_{i=1}^n r_i \sum_{i=1}^n d_i^2 - \sum_{i=1}^n d_i \sum_{i=1}^n d_i r_i \right]}{\left[ n \sum_{i=1}^n d_i^2 - \left( \sum_{i=1}^n d_i \right)^2 \right]} \quad (5)$$

Table 1: The eight isometric transformations

Sym.	Angles	Equations	Results	Sym. Equations
				R=Range block BS=Block Size
$T_0$	Identity	$x' = x \cos(0) + y \sin(0)$ $y' = -x \sin(0) + y \cos(0)$	$x' = x$ $y' = y$	$Sym(x,y)=R(x,y)$
$T_1$	Rot.(+90)	$x' = x \cos(90) + y \sin(90)$ $y' = -x \sin(90) + y \cos(90)$	$x' = y$ $y' = -x$	$Sym(x,y)=R(y,BS-x)$
$T_2$	Rot.(+180)	$x' = x \cos(180) + y \sin(180)$ $y' = -x \sin(180) + y \cos(180)$	$x' = -x$ $y' = -y$	$Sym(x,y)=R(BS-x,BS-y)$
$T_3$	Rot.(+270)	$x' = x \cos(270) + y \sin(270)$ $y' = -x \sin(270) + y \cos(270)$	$x' = -y$ $y' = x$	$Sym(x,y)=R(BS-y,x)$
$T_4$	Ref. at x-axis	$x' = -x \cos(0) + y \sin(0)$ $y' = -x \sin(0) + y \cos(0)$	$x' = -x$ $y' = y$	$Sym(x,y)=R(BS-x,y)$
$T_5$	Ref. & Rot. (90)	$x' = -x \cos(90) + y \sin(90)$ $y' = -x \sin(90) + y \cos(90)$	$x' = -y$ $y' = -x$	$Sym(x,y)=R(BS-y,BS-x)$
$T_6$	Ref. & Rot.(180)	$x' = -x \cos(180) + y \sin(180)$ $y' = -x \sin(180) + y \cos(180)$	$x' = x$ $y' = -y$	$Sym(x,y)=R(x,BS-y)$
$T_7$	Ref. & Rot. (270)	$x' = -x \cos(270) + y \sin(270)$ $y' = -x \sin(270) + y \cos(270)$	$x' = y$ $y' = x$	$Sym(x,y)=R(y,x)$

In order to show the huge number of computations needed in the full search of BFIC, the following example is illustrated, if  $f$  is a 256 x 256 image and the range block size is 4, the range pool  $R$  is composed of  $(256/4) \times (256/4) = 4096$  blocks of size 4x4 and the domain pool  $D$  is composed of  $(256-8+1) \times (256-8+1) = 62001$  blocks and each domain block must be test in all its 8 isomeric transformation so the total number of domain blocks in the domain pool will  $62001 \times 8 = 496008$ . Thus for each rang block in the range pool, there are 496008 MSE computations must be done in order to obtain the most similar block from the domain pool. Thus, in total, there are  $4096 \times 496,008 = 2,031,648,768$  MSE computations needed to encode the whole image using BFIC full search compression method. So this work aimed to speed-up the BFIC by reducing the number of these computations.

The result of the FIC will be a set of transformations ( $w$ ); the number of these transformations is equal to the number of range blocks ( $r$ ) in the range pool  $R$ . For each transformation to be encoded, 31 bits are needed (8 bits in each of the  $x$  and  $y$  directions to determine the position of  $D_i$ , 7 bits for  $o_i$ , 5 bits for  $s_i$ , and 3 bits to determine a rotation and flip operation for mapping  $d_i$  to  $r_i$  i.e. the 8 isometric cases of table 1) [34].

Decoding process is considerably easier and faster than the encoding process, starting from any initial image, for each  $w_i$  it found  $d_i$  for the corresponding range  $r_j$ , multiply the pixel values by  $s_i$  and add to  $o_i$  put the resulting pixel values in the position of  $r_j$  see Eq. (6) [35].

$$\tilde{r} \approx S \cdot d_i + o \quad (6)$$

This process will be iterated until an approximation to the fixed point (attractor) is reached. Typically, 8 iterations are sufficient [28].

#### 4. Proposed Method

The proposed method AFIC aimed to develop the performance of BFIC in terms of the Encoding Time (ET), Compression Ratio (CR) and the reconstructed image quality that measured by Peak Signal to Noise Ratio (PSNR) and make a good trade off among them, for that AFIC used the following techniques:

##### 4.1 Adaptive Quadtree Partitioning Technique (AQPT)

The block size  $B$  is an essential parameter in BFIC because it is hard to find a proper domain block that can be efficiently mapped to a large range block, and this reduces the reconstructed image quality but it increases the CR and reduces the ET. On the other hand, if the range block size was small, a much better reconstructed image quality can be obtained but the CR will be decreased and the ET will be increased significantly. (It is easy to verify that the compression ratio is proportional to the value of  $B^2$ ) [7]. The hierarchical partitioning techniques like the quadtree are adaptive partitioning schemes that take the difficulties and the natural connections between areas into account, so the image to be partitioned will be partitioned into regions (i.e. ranges) of different size depending on the content of the image region, smaller ranges for the regions that have more details and larger ranges for regions having simple details and this will lead to a good trade off among the ET, CR and the PSNR by:

a- Reduce the number of the ranges that have no important details and then reduce the encoding time and increase the compression ratio.

b- Concern on the regions that have more important details and then give a good reconstructed image quality.

Unlike the quadtree partitioning technique used by Fisher [34], aiming to simplify and then speed-up the BFIC encoding stage, at the AQPT used in this work, the image partitioning will be done firstly as a separated step from the matching process [36].

At AQPT, The ranges will be created using algorithm consists of following steps:

1. Assume that the whole input image is defined as only one range block  $r_0$ .
2. Compute the global mean, the global squared mean and the standard deviation ( $\sigma$ ) of the input image (i.e., for  $r_0$ ) using the following equations:

$$\bar{M} = \frac{1}{W \times H} \sum_{y=1}^W \sum_{x=1}^H R_0(x, y) \quad (7)$$

$$\bar{M}^2 = \frac{1}{W \times H} \sum_{y=1}^W \sum_{x=1}^H (R_0(x, y))^2 \quad (8)$$

$$\sigma = \sqrt{\bar{M}^2 - (\bar{M})^2} \quad (9)$$

( $\bar{M}$ ) and  $\sigma$  are measures of distribution of the pixel values in the image. The extended standard deviation ( $\sigma_E$ ) is used as a tolerance criterion. It is computed as multiples of standard deviation.

3. Select control parameters of partitioning process to decide the image block to be partitioned or not. These parameters are:

- I) **Maximum Block Size (*MaxSiz*):** Represents the maximum allowable size of the range block which corresponds to the minimum depth of the partitioning tree.
- II) **Minimum Block Size (*MinSiz*):** Represents the minimum allowable size of the range block which corresponds to the maximum depth of the partitioning tree.
- III) **Inclusion Factor (*I<sub>f</sub>*):** Represents the multiple factor, when it is multiplied by the global standard deviation ( $\sigma$ ), it will define the value of  $\sigma_E$ . The selection of small  $I_f$  value will lead to high quality and low compression ratio of reconstructed image, and vice versa.

IV) **Acceptance Ratio ( $R_a$ )**: Represents the ratio of the number of pixels whose gray values differ from the block mean by a distance farther than the expected  $\sigma_E$  value. As the  $R_a$  value is selected small it can get a higher quality of the reconstructed image, and vice versa.

4. A linked list has been designed to store all information about quadtree partitioning process. This linked list is defined as an array of records. Each record consists of the following fields; **Position**: Represents the upper left coordinates of each image block (i.e.  $x$  and  $y$ ), **Size**: Represents the size of each image block, **Next**: It is a pointer to the next range block.

Quadtree partitioning process usually starts with partitioning the image into blocks whose size is equal to the maximum block size. If any sub-block satisfies the uniformity condition then the block is accepted and stored, otherwise, the block should be partitioned into four sub-blocks. Each block should be examined by measuring its uniformity. If it does not satisfy the uniformity criterion, partitioning should be proceeded. The partitioning process is repeated until the uniformity condition is satisfied.

Two parameters are used to control uniformity function, which are the Inclusion Factor ( $I_f$ ) that is used to adjust the extended standard deviation ( $\sigma_E = I_f \sigma$ ) (Extended standard deviation is used as a metric for the degree of minimum allowed dispersion in the spatial distribution of the gray values). The other parameter is the Ratio ( $R_a$ ). This factor is used as a tolerance for relative number of pixels within the block, which may differ from the mean value of the block by distance of more than  $\sigma_E$  [36].

In figure 3, 256x256 Lena image is partitioned using AQPT with different partitioning parameters values. Different parameters values led to different number of range blocks and this in turn affects the CR, ET and the PSNR.

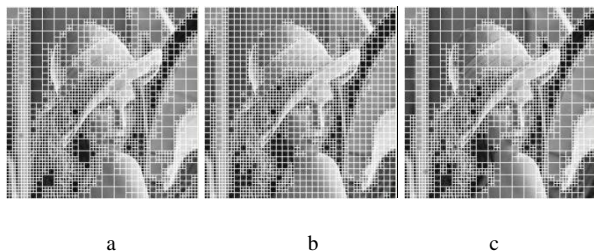


Fig. 3 a: MaxSiz=16, MinSiz=4, If =0.5, Ra=0.02 (2455 range blocks)  
 b: MaxSiz=8, MinSiz=4, If =0.7, Ra=0.03 (2377 range blocks)  
 c: MaxSiz=16, MinSiz=4, If =0.8, Ra=0.02 (1987 range blocks)

## 4.2 Zero Mean Intensity Level FIC (ZMIL FIC)

ZMIL introduces the transforms of the full search problem using a more convenient form by adopting the mean of the range block  $\bar{r}$  as parameter that has better properties than the conventional offset parameter  $o$  and a new search algorithm has been developed. In fact, this idea was advocated by Oien and Lepsoy and also implicitly used by Bani-Egbal and it was applied by Tong and Pi [37]. And in this research it was used and developed by merging it with other speeding up techniques to get a hi-performance method. As illustrated in our previous work [25] ZMIL can convert the full search scheme to a one-parameter optimization problem since the optimal approximation in the decoding unit for every range block can be obtained from Eq. (10).

$$r_i = s(d_i - \bar{d}) + \bar{r} \quad \forall i \quad (10)$$

Where:

$\bar{r}$  : represents the average (mean) for a specific range block.

$\bar{d}$  : represents the average (mean) for the mapped domain block for this range block.

From Eq. (10), it is noticed that the fractal parameter  $\bar{r}$  is used instead of the conventional  $o$  coefficient. So ZMIL FIC parameters will be:  $\bar{r}$  (DC component of the range) which is independent of the domain block and  $s$  which is related to the (AC-component) of the range block [37]. Using  $\bar{r}$  as one of the affine parameters instead of the  $o$  will lead to:

- Decoupling of the optimization of the two affine parameters and thus furthering speed up the search for the best matching domain block [32], [37].
- It is more efficient to quantize  $\bar{r}$ , especially as it has a much smaller dynamic range [0, 255], than the  $o$  parameter [-255, 510] [37][40]. So, it is more cost effective (in terms of minimizing the quantization error per code).
- In the introduced transformation,  $\bar{r}$  is uniformly quantized by 6 bits and  $s$  is uniformly quantized by 2 bits instead of 5 and 7 bits for  $s$  and  $o$  in BFIC and this will lead to more CR [25].

The distortion equation used by ZMIL will be:

$$E(R', D') = \frac{1}{n} \sum_{i=1}^n (sD'_i - R'_i)^2 \quad (11)$$

Where

$$R' = r_i - \bar{r}$$

$$D' = d_i - \bar{d}$$

And this means that the range-domain blocks have been adjusted to a Zero Mean Intensity Level (ZMIL) by subtracting the mean  $(\bar{r}, \bar{d})$  from all the range-domain blocks. Minimizing Eq. (11) by derivative of  $E(R', D')$  with respect to  $s$  as zero, i.e.,

$$\frac{\partial E(R', D')}{\partial s} = \frac{-2}{n} \sum (sD'_i - R'_i) D'_i = 0$$

$$\sum (R'_i D'_i) = s \sum D_i'^2$$

$$\therefore s = \frac{\sum R'_i D'_i}{\sum D_i'^2} \quad (12)$$

In practice a significant improvement in fidelity can be obtained if the quantized  $s$  value is used in computing the err  $E(R', D')$  during encoding process, where the quantity that we actually want to minimize is the root mean square error  $\sqrt{E(R', D')}$  where  $E(R', D')$  is:

$$E(R', D') = \frac{1}{n} [s^2 \sum D_i'^2 + \sum R_i'^2 - 2s \sum R'_i D'_i] \quad (13)$$

### 4.3 Reducing the Domain Image Size (RDIZ)

This technique works on minimizing the domain pool; as mentioned previously in BFIC, the encoding process is computationally intensive. A large number of sequential searches through a list of domains are carried out while trying to find a best match for a range block.

A large domain pool will increase the number of comparisons that have to be made to find the best domain block and this where most of the computing time is used [38].

To avoid doing any repeated spatial contraction of the domain blocks at the domain pool generation stage, in RDIZ the entire image is reduced to  $1/16^{\text{th}}$  of its original size at the start of the compression operations, and the domain blocks are then chosen from the reduced image with no need for further scaling. This way of creating the domain has an important point; it greatly decreases the computational process by reducing the number of domain blocks by a factor of sixteen by down sampling every  $4 \times 4$  (instead  $2 \times 2$ ) pixels in the original image (using the average method) to one pixel in the reduced domain image, see Figure 4. In this case, for the image  $f$  in our previous example and if the domain jump step is 4 (as it is used in our experimental results), the number of the domain blocks needed in the match process for each range block will be reduced to only 256 domain blocks. So, the computations needed in the encoding process will be reduced to only  $(4096 \times 256 \times 8) = 8,388,608$  and this will lead to a significant reduction in the encoding time.

Also RDIZ will reduce the number of bits required to encode the original image because the number of bits needed for storing each of  $x$  and  $y$  coordinates of the best matched domains will be decreased. In our example the domain pool will be of size  $(64 \times 64)$  pixels so the maximum value for each  $x$  and  $y$  coordinates will be (60) by dividing it on the jump step  $(60/4 = 15)$  then the encoder will need 4 bits to store each of  $x$  and  $y$  coordinates instead of 8 bits needed in the case of the full search in BFIC. Accordingly, this will lead to remarkable increase in the compression ratio.

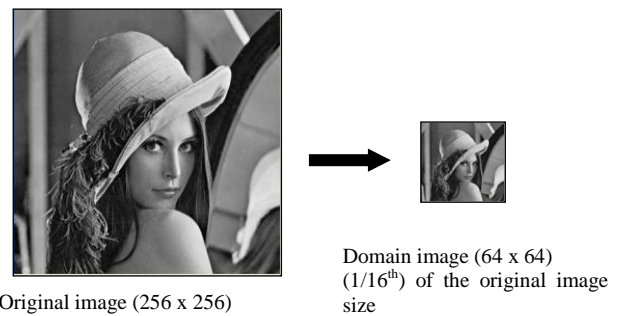


Fig. 4 Creating the domain pool of Lena image by down sampling the image using the average of  $(4 \times 4)$  pixels.

#### 4.4 Range Exclusion (RE)

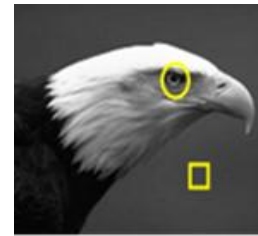
Large efforts have been undertaken to speed up the encoding process. Most of the proposed techniques attempt to accelerate the searching and are based on some kind of feature vector assigned to ranges and domains. A different route to increased speed can be chosen by less searching as opposed to faster searching [39], [38]. RE technique worked to speed-up the BFIC by reducing the number of range blocks required in matching process through excluding homogenous ranges from the search process. For that, the mean and standard variance features of all range blocks will be extracted. The mean value  $\bar{r}$  (Eq. 14) of a range  $r$  gives the measure of its average gray level; and the standard variance ( $Vr$ ) (Eq. 15) value of range ( $r$ ) defines the dispersion of its gray level from its mean. So,  $Vr$  will be used to check the range ( $r$ ) whether is it homogenous (flat) region or not? In homogenous region the value of the  $Vr$  is about zero while it increases in the areas with more details. The flat region means that all pixels of this region have the same value or are close to each other so it can be substituted by its mean value  $\bar{r}$  [38]. Figure 5 shows an example of homogeneous and detailed regions that can be found in an image.

After the partitioning process, the homogeneity of each range will be checked before starting the matching process. During the matching process, the homogeneous ranges will be excluded. So, the matching operation will be limited to the detailed regions only and this will lead to avoid a large amount of complex calculations, which results as a fast coding process. In order to achieve a better performance, the homogeneity criterion is controlled by using different values of the  $Vr$ ; for that a threshold called Homogenous Permittivity ( $HP$ ) is used which represents the amount of homogeneity allowed. If the variance of any range is less than or equal to the  $HP$  value, the range will be excluded from the search and matching operation and it will be encoding only by saving its mean value. So this process will speed-up the BFIC significantly and also a higher compression ratio will be achieved because each excluded range will require only 6 bits to store its quantized mean value as its fractal code instead of the 31 bits required to store its IFS code parameters in the full search of the BFIC.

$$\bar{r} = \frac{1}{X_{size} \times Y_{size}} \sum_{i=0}^{X_{size}-1} \sum_{j=0}^{Y_{size}-1} X_{ij} \quad (14)$$

$$Vr = \frac{1}{X_{size} \times Y_{size}} \sum_{i=0}^{X_{size}-1} \sum_{j=0}^{Y_{size}-1} [X_{ij} - \bar{r}]^2 \quad (15)$$

Fig. 5 the box is a homogenous region, while the circle region contains different combinations



#### 4.5 Variance Domain Selection (VDS)

This technique is responsible for reducing the number of domain blocks that required to be searched for each not homogenous range block. For that, there are three steps in the VDS:

- 1- Select only the domains blocks that have a variance  $Vd$  close to the range  $Vr$  to be searched. As we explained in section 4.4, the variance is a criterion for the level of the range details (i.e. homogenous, has simple details or complex details) so, for example, it is not reasonable to search flat domain blocks for a complex range block and vice versa. For that, a threshold called  $SD_{Th}$  is used, and only the domain blocks that satisfy the condition of the Eq. 16 will be selected to be searched in the matching process so the domain blocks needed to be searched will be reduced and then the ET will be reduced significantly.

$$\text{if } |Vr - Vd| \leq SD_{Th} \quad (16)$$

- 2- Use a stopping condition for the search process based on the monitoring the minimum matching error. The most direct and easy way to reduce the search complexity is by monitoring the matching error [40]. At any matching instance, the error is checked, if it is below a pre-defined permissible level (threshold)  $TH_E$  then the registered domain block is considered as the best matched block and, then, the search across the domain blocks is stopped. And this will reduce the required long fractal coding time.
- 3- Additional speeding-up can be achieved by taking only the first four symmetry transform cases ( $T_0$ ,  $T_1$ ,  $T_2$  and  $T_3$  of table 1). This will reduce the number matching operations and then the encoding time significantly. Also a more compression ratio can be achieved since there are only four symmetry cases and this will need only 2 bits instead of 3 bits to be coded but with some drawback in the reconstructed image quality so in some applications, when a high

reconstructed image quality is needed, the full 8 symmetry cases will be used. Algorithms 1 and 2 show the steps of the AFIC encoding and decoding stages respectively.

**Algorithm 1 the AFIC encoding stage**

Input: The original image

Output: The IFS code

Method:

Step1: Load the image into an *Orimage* array

Step2: Initialize all the encoding parameters i.e. *MaxSiz*, *MinSiz*, *I<sub>f</sub>*, *R<sub>a</sub>*, *HP*, *SD<sub>Th</sub>*, *TH<sub>E</sub>*.

Step2: Partitioning the image into non-overlapped range blocks (*r<sub>1</sub>... r<sub>n</sub>*) of different size using the AQPT

Step3: Generate the minimized domain image from the original image using RDIZ

Step4: Build a new domain blocks (*d'<sub>1</sub>...d'<sub>m</sub>*)

Step5: For each range block *r<sub>i</sub>* compute its mean value  $\bar{r}_i$  (Eq.14) and variance value *Vr<sub>i</sub>* (Eq.15) and check:

If  $Vr_i \leq HP$  then quantize  $\bar{r}_i$  and store it as its FIC code and move to the next range.

else do:

- i. Quantize  $\bar{r}$
- ii. Through all the *D'* select only the domain blocks that satisfy the condition of Eq. 16: to be searched for the best matching *d'<sub>i</sub>* so:

if  $|Vr - Vd| \geq SD_{Th}$  leave *d'<sub>i</sub>* and try the next *d'*

else for all possible symmetry cases *sym* do:

- Compute *s* using Eq. 12
- Quantize the *s*;
- Compute  $E(R'_i, D'_i; s)$ ;
- Compare the result *E* with the minimum *E* registered during the previous matching instances. If *E* is smaller then put its value in minimum *E* register (beside to the associated values of *sq*, *sym*, *xd*, *yd*). In case of the new registered minimum *E* is less than the permissible level of error *TH<sub>E</sub>* then stop the search process, and output the set (*xd*, *yd*, *sq*,  $\bar{r}_q$  and *sym*) as best IFS match for the tested range block else go to the next domain block.

**Algorithm 2 the AFIC decoding stage**

Input: The IFS code

Output: The decoded image

Method:

Step1: Generate the first minimized domain image arbitrary with 1/16<sup>th</sup> of the original image size.

Step2: Determine the iterations number

Step3: Load IFS code

Step4: Dequantize the value of scale *s<sub>i</sub>* and range block mean  $\bar{r}_i$

Step5: Build a new domains blocks (*D'<sub>1</sub>...D'<sub>m</sub>*)

Step6: Reconstruct the range block as the following:

Check the contents of the IFS code;

if the IFS code has the mean value only this means that the range is homogenous and it will be reconstructed from its quantized mean value only.

$$R_i = \bar{r}$$

else (IFC contains all the related code i.e *xd*, *yd*, *sq*,  $\bar{r}_q$  and *sym*) then the range will reconstructed using Eq. 10.

Step7: Each range block is reconstructed will be located in its position in the decoded image plane.

Step8: Down sample the decoded (reconstructed) image into the size of minimized domain image by averaging using RDIZ.

Step9: Repeat from step 5 until the attractor state is reached (i.e., decoded image will not be changed as we processed in the iterations)

**5. Experimental Results**

The proposed method AFIC is simulated to verify its performance. The tested images are Lena, Pepper and Zelda. All the images are gray scale of size 256 × 256 pixels. The method was programmed using visual C++ and implemented on a Pentium Dual Core 3.8 GHz, Windows XP PC.

The results showed that varying values of the AFIC parameters (*MaxSiz*, *MinSiz*, *I<sub>f</sub>*, *R<sub>a</sub>*, *HP*, *SD<sub>Th</sub>*, *TH<sub>E</sub>*, domain jump step *j*, the number of bits allocated for the contrast scaling factor *ScIBits* and the number of bits allocated for the mean  $\bar{r}$  of range block *MrBits*) lead to different performance results.





We found that a best tradeoff among the ET, CR and the reconstructed image quality (PSNR) can be achieved with the following parameters values:

*MaxSiz* = 16 or 8, *MinSiz* = 4, *I<sub>f</sub>* = 0.1-0.9, *R<sub>a</sub>* = 0.01-0.09, *HP* = 10-75, *SD<sub>Th</sub>* = 10-50, *TH<sub>E</sub>* = 0.4 x range size, *j* = 2 or 4, *ScIBits* = 2 or 3, *MrBits* = 6.



Table 2 shows some of the experimental results on Lena image.

Table 2: Some of experimental results on Lena image with  $SclBits = 3$ ,  $MrBits = 6$ ,  $j = 4$  and different values for other parameters

Parameters Value		Performance Results		The Reconstructed Image
Parameter	Value	Criterion	Value	
Test 1				
<i>MaxSiz</i>	16	Ranges No	2563	
<i>MinSiz</i>	4	excluded ranges	587	
<i>I<sub>f</sub></i>	0.5	ET (Sec.)	1.61	
<i>R<sub>a</sub></i>	0.01			
<i>HP</i>	35	CR	10.88	
<i>SD<sub>Th</sub></i>	50			
<i>Sym Cases</i>	Full 8	PSNR	30.40	
Test 2				
<i>MaxSiz</i>	8	No. of Ranges	2731	
<i>MinSiz</i>	4	No. of excluded ranges	266	
<i>I<sub>f</sub></i>	0.5	ET (Sec.)	1.89	
<i>R<sub>a</sub></i>	0.01			
<i>HP</i>	10	CR	9.72	
<i>SD<sub>Th</sub></i>	10			
<i>Sym Cases</i>	First 4	PSNR	30.34	
Test 3				
<i>MaxSiz</i>	16	No. of Ranges	2563	
<i>MinSiz</i>	4	No. of excluded ranges	626	
<i>I<sub>f</sub></i>	0.4	ET (Sec.)	1.38	
<i>R<sub>a</sub></i>	0.01			
<i>HP</i>	30	CR	10.41	
<i>SD<sub>Th</sub></i>	10			
<i>Sym Cases</i>	full 8	PSNR	30.05	
Test 4				
<i>MaxSiz</i>	8	No. of Ranges	1762	
<i>MinSiz</i>	4	No. of excluded ranges	541	
<i>I<sub>f</sub></i>	0.9	ET (Sec.)	0.75	
<i>R<sub>a</sub></i>	0.08			
<i>HP</i>	60	CR	17.47	
<i>SD<sub>Th</sub></i>	15			
<i>Sym Cases</i>	First 4	PSNR	28.25	

To evaluate the proposed method, AFIC results are compared with BIFC full search method, Duh's classification method [16], Tseng's PSO-KI method [13], and EP-NRS method [24]. Table 3 shows the comparison results of the compared methods on Lena 256 x 256 gray scale image, where the number of classes is 55 for Duh's method, particle population size and number of rounds are 35, 33 respectively for PSO-KI method, the value of  $(\hat{\theta}, c)$  pair are  $(3.2^\circ, 0.44)$  respectively for EP-NRS (for more information about these method parameters see [16], [13] and [24]). The range block size is 8x8 for all of the BIFC full search, Duh's method, Tseng's PSO-KI method and Yih's EP-NRS method. The parameters values of the proposed method (AFIC) are the same of test 4 in table 2. The results are taken under the condition of about the same value of PSNR.

Table 3: Comparisons results among BIFC, Duh's method, PSO-KI, EP-NRS and AFIC on 256x256 Lena image

Comp. Method	PSNR (dB)	ET (s)	Speedup ratio	No. of MSE computations	Reduce d ratio	CR
BIFC Full Search	28.91	770.31	1.0	475,799,552	1.0	16.13
Duh	28.01	16.26	47.37	8,650,920	55.0	16.13
PSO-KI	27.98	6.10	126.28	3,496,880	136.06	16.13
EP-NRS	28.01	6.37	121.30	3,949,372	120.47	16.13
AFIC	28.25	0.75	1027.08	1,177,140	404.20	17.47

Comparing to the full search of BIFC; The comparisons show that:

Duh's method achieved; About 55 times reduction in the total number of MSE computation and about 47 speedup ratio with 0.9 dB decay in the reconstructed image quality.

PSO-KI method achieved; About 136 times reduction in the total number of MSE computation and about 126 speedup ratio with 0.93 dB decay in the reconstructed image quality.

EP\_NRS method achieved; About 120 times reduction in the total number of MSE computation and about 121 speedup ratio with 0.9 dB decay in the reconstructed image quality.





The proposed method (AFIC) achieved; About 404 times reduction in the total number of MSE computation and about 1027 speedup ratio with only 0.66 dB decay in the

reconstructed image quality. Moreover AFIC gets a more compression ratio than others, it get 1.34 more compression ratio to become 17.47 while all other methods got only 16.13 compression ratio.

So AFIC indeed has a better performance than the compared methods since it used many techniques concerned on reducing the ET and increasing the CR at the same time (i.e. AQPT, RE and RDIZ). Also ZMIL helped to; reduce the complexity of the matching operations and then reduces the ET, increase the CR and the reconstructed image quality.

Table 4 shows some the test results on Peppers and Zelda 256x256 images.

Table 4: Some of experimental results on Peppers and Zelda image with  $SclBits = 3$ ,  $MrBits = 6$ ,  $j = 4$  and different values for other parameters

Parameters Value		Performance Results		The Reconstructed Image
Parameter	Value	Criterion	Value	
The same values of test 1 in table 2		R. No.	2284	Test 1  Peppers
		E.R.	593	
		ET (s)	1.51	
		CR	12.49	
		PSNR	30.27	
The same values of test 2 in table 2		R. No.	2515	Test 2  Peppers
		E.R.	356	
		ET (s)	1.01	
		CR	10.87	
		PSNR	30.15	
The same values of test 3 in table 2		R. No.	3172	Test 3  Zelda
		E.R.	961	
		ET (s)	1.62	
		CR	9.30	
		PSNR	34.65	
The same values of test 4 in table 2		R. No.	1492	Test 4  Zelda
		E.R.	434	
		ET (s)	0.70	
		CR	20.37	
		PSNR	31.13	

## 6. Conclusions

In this paper, an Adaptive Fractal Image Compression AFIC is proposed, AFIC aimed to; decrease the encoding time and increase the compression ratio at the same time. Also it made to keep the reconstructed image quality as much as possible. For that, AFIC used AQPT, ZMIL, RE, RDIS and VDS techniques that illustrated in sections 4.1-4.5, all these techniques worked together to achieve the aim of AFIC and to make a good trade off among the ET, CR and the PSNR. AFIC has many control parameters, these parameters give AFIC a good flexibility to get different results, i.e. varying the parameters values led to variant results in term of CR, ET and PSNR so we can get a higher PSNR on account on ET and CR and vice versa to accomplish the requirements of different applications. The compression results show that AFIC has better performance than BFIC, Duh's method, PSO-KI and EP-NRS, AFIC is about 21.68, 8.13 and 8.49 times faster than Duh's method, PSO-KI and EP-NRS methods respectively also it gets 1.34 CR more than all other compared methods. In term of the reconstructed image quality also we find that in AFIC, the penalty of retrieved image quality only decays by 0.66 dB while it decays by 0.9, 0.93 and 0.9 dB in Duh's method, PSO-KI and EP-NRS methods respectively.

## Acknowledgement

This work was supported by the Natural Science Foundation of China (NSFC) (No. 60873140, 61073125 and 61071179), the Program for New Century Excellent Talents in University (No. NCET-08-0155 and NCET-08-0156), and the Fok Ying Tong Education Foundation (No. 122035).

## References

- [1] Y.Chakrapani, and K.Soundera Rajan, "Hybrid Genetic-Simulated Annealing Approach for Fractal Image Compression", International Journal of Information and Mathematical Sciences, Vol.4, No.4, 2008, pp. 308-313.
- [2] L. Torres, and M. Kunt, Video Coding: The Second Generation Approach, Boston: Kluwer Academic Publishers, 1996.
- [3] W. Xing-Yuan, and et al, "Fractal image compression based on spatial correlation and hybrid genetic algorithm", Journal of Visual Communication and Image Representation, Vol.20, No.8, 2009, pp. 505-510.
- [4] M.F. Barnsley, Fractals Everywhere, New York: Academic, 1988.
- [5] M.F. Barnsley, and A.D. Sloan, "A better way to compress images", BYTE Magazine, V.13, No.1, 1988, 215-233.
- [6] K. Jaferzadeh, and et al, "Acceleration of fractal image compression using fuzzy clustering and discrete-cosine-transform-based metric", IET Image Process, Vol.6, No.7, 2012, pp. 1024-1030.

- [7] T. Kovács, "A fast classification based method for fractal image encoding", *Image and Vision Computing*, Vol.26, No.8, 2008, pp. 1129-1136.
- [8] A.E. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations", *IEEE Transactions on Image Processing*, Vol.1, No.1, 1992 pp. 18–30.
- [9] H.O. Peitgen, and et al, *Fractals in the Fundamental and Applied Sciences*, New York: Elsevier Science Publishing Company Inc., 1991.
- [10] J. Crilly, and et al, *Fractals and Chaos*, New York: Springer-Verlag, 1991.
- [11] V. Chaurasia, and A. Somkuwar, "Speed up Technique for Fractal Image Compression", in *International Conference on Digital Image Processing*, 2009 pp. 319-323, "doi. 10.1109/ICDIP.2009.66".
- [12] X. Y. Wang and et al, "An improved fast fractal image compression using spatial texture correlation", *Chin. Phys. B*, Vol. 20, No. 10, 2011, 104202.
- [13] C.C. Tseng, and et al, "Fractal image compression using visual-based particle swarm optimization", *Image and Vision Computing*, Vol. 26, No.1, 2008, pp. 1154–1162.
- [14] Z. Wang, and et al, "Hybrid image coding based on partial fractal mapping", *Signal Processing: Image Communication*, Vol. 15, No. 9, 2000, pp. 767–779.
- [15] D.J. Duh, and et al, "DCT based simple classification scheme for fractal image compression", *Image and Vision Computing*, Vol. 23, No. 13, 2005, pp. 1115–1121.
- [16] D.J. Duh, and et al, "Speed quality control for fractal image compression", *The Imaging Science Journal*, Vol. 56, No.2, 2008, pp. 79–90.
- [17] S. Furao, and O. Hasegawa, "A fast no search image coding method", *Signal Processing: Image Communication*, Vol.19, No.5, 2004, pp. 393–404.
- [18] K.L. Chung, and C.H. Hsu, "Novel prediction- and subblock-based algorithm for fractal image compression", *Chaos, Solitons & Fractals*, Vol. 29 No. 1, 2006, pp. 215–222.
- [19] X.Y. Wang, and S.G. Wang, "An improved no-search fractal image coding method based on a modified gray-level transform", *Computers & Graphics*, Vol. 32, No. 4, 2008, pp. 445–450.
- [20] T.K. Truong, and et al, "Fast fractal image compression using spatial correlation", *Chaos, Solitons & Fractals*, Vol. 22, No.5, 2004, pp. 1071–1076.
- [21] M.S. Wu, and et al, "Spatial correlation genetic algorithm for fractal image compression", *Chaos, Solitons & Fractals*, Vol. 28, No. 2, 2006, pp. 497–510.
- [22] Y. Zheng, and et al, "An improved fractal image compression approach by using iterated function system and genetic algorithm", *Computers & Mathematics with Applications*, Vol. 51 No. 11, 2006, pp. 1727–1740.
- [23] Y.-L. Lin, and M.-S. Wu, "An edge property-based neighborhood region search strategy for fractal image compression", *Computers and Mathematics with Applications*, Vol. 62, No. 1, 2011, pp. 310–318.
- [24] T. Hasan, and et al, "Two Suggested Methods for Faster Fractal Image Compression", *Research Journal of Applied Sciences, Engineering and Technology*, Vol. 3, No. 8, 2011, pp. 757-764.
- [25] <http://en.wikipedia.org/wiki/Self-similarity>, 1/6/2011.
- [26] E. Vrcasy, and L. Colin, "Image Compression Using Fractals", *IBM Journal of Research and Development*, Vol. 65, No.19, 1995, pp. 121-134.
- [27] S. Abdul-Khalik, "Fractal Image Compression Using Shape Structure", M.Sc. thesis, College of Science, Al-Mustansiriya University, 2005, Iraq.
- [28] X.Y. Wang, and et al, "An effective fractal image compression algorithm based on plane fitting", *Chin. Phys. B*, Vol. 21, No. 9, 2012, 090507.
- [29] P. Xiao, "Image Compression by Wavelet Transform", M.Sc. thesis, College of Science, East Tennessee State University, 2001.
- [30] H.R. Mahadevaswamy, "New Approaches to Image Compression", Ph.D. thesis, Regional Engineering College, university of Calicut, 2000, India.
- [31] S. Lee, "Parallel Processing Architecture for Fractal Image Compression", Ph.D. thesis, College of engineering, Tohoku University, 2000, Japan.
- [32] E. Abdul Malik, "Speeding-Up Fractal Colored Image Compression using Moment Features", Ph.D. thesis, College of Science, Al-Mustansiriyah University, Baghdad, 2007, Iraq.
- [33] Y. Fisher. (Ed.), "Fractal Image Compression: Theory and Applications", New York, NY, USA, Springer, 1995.
- [34] E. Saad, "Suggested Algorithm for Fractal Image Compression", M. Sc. Thesis, College of Science, Al-Mustansiriya University, 2006, Iraq.
- [35] H. S. Jamila, "Fractal Image compression", Ph.D. thesis, College of Science, University of Baghdad, 2001, Iraq.
- [36] C.S. Tong, and M. Pi, "Fast Fractal Image Encoding Based on Adaptive Search", *IEEE Trans. Image Process*, Vol. 10, No. 9, 2001, pp. 1269-1277.
- [37] T. Hasan, and X. Wu "An Adaptive Algorithm for Improving the Fractal Image Compression (FIC)", *JOURNAL OF MULTIMEDIA*, VOL. 6, NO. 6, 2011, pp 477-485.
- [38] D. Saupe, "Accelerating Fractal Image Compression by Multi-Dimensional Nearest Neighbor Search", *CC'95 Data Compression Conference*, J. A. Storer, M. Cohn (eds.), IEEE Computer Society Press, 1995, pp. 222-231.
- [39] E. George, "IFS Coding for Zero-Mean Image Blocks", *Iraqi Journal of Science*, Vol.47, No.1, 2006, PP. 190-194.

**Taha Mohammed Hasan** received his BSc, MSc in computer science from Mansour University College and The University of Mustansiriyah, Baghdad, Iraq in 1992 and 2006 respectively. He is currently pursuing the Ph.D. degree at the Harbin Institute of Technology (HIT), Harbin, China. His research interests is the image processing.

**Xinagqian Wu** received his B.Sc., M.Sc. and Ph.D. in computer science from Harbin Institute of Technology (HIT), Harbin, China in 1997, 1999 and 2004, respectively. Now he is a professor in the School of Computer Science and Technology, HIT. His current research interests include pattern recognition, image analysis and biometrics, etc.