A Speeding Up Fractal Image Compression Using Fixed Size Partition and Hierarchical Classification of Sub-images

SWALPA KR. ROY  S. K. BANDYOPADHYAY  ABHISHEK MAHATO  TAI - HOON KIM ∗
IIEST, Shibpur  University of Calcutta  IIEST, Shibpur  Sungshin Women’s University
India - 711103  India - 711103  India - 711103  Korea-136742
swalpa@students.becs.ac.in  skb1@vsnl.com  abhimahatu@gmail.com  taihoonn@daum.net

Abstract: Fractal Image Compression (FIC) is one of the best techniques for natural and still images. In this method an image is divided into non-overlapping range blocks and overlapping domain blocks. The total number of domain blocks is larger than the range blocks. Similarly the size of the domain blocks are larger than the range blocks. Together all domain blocks creates a domain pool. A range is compared with all possible domains for similarity measure. However this process is very time consuming. The size of the domain pool dominates the fractal encoding phase. In this paper, a novel speeding up fractal image compression technique is proposed which combine fixed size partition of image and a two level classification of domain/range blocks to reduced the number of domain blocks so on range blocks. A range is compared with only those domains which are in the same level of the classification group. Experiments on some standard image data show that the proposed technique numerous reduces the compression time when compared with baseline fractal image compression (BFIC) and comparable with other scheme proposed till date.

Key–Words: Fractal Image Compression, Fisher’s Classification, Hierarchical Classification, DCT, IFS, PIFS

1 Introduction

A major objective of image coding is to represent digital images with as few bits as possible while preserving the level of intelligibility, usability or quality required for the application. Image compression has two great purposes: reduction of time for image transmission and reduction of storage area. The level of information quality required vary widely, depending on the special tasks. Several are the forms used to compact data, many of them can be used to compress images if after compression and reconstruction it must be exactly equal to the original[1]. Image coding techniques which do not destroy any information and which allow exact reconstruction of the original digital data are said to be information-preserving (or lost free). In such applications as image analysis from space programs and medical diagnosis, exactly regeneration of the original data is expected.

Fractal Image Compression is a lossy image compression technique for digital images, based on the theory of fractals proposed by Michael F. Barnsley (1988), using Iterated Function Systems (IFS)[2, 3]. This method exploits the fact that real world images are highly self-similar i.e. different portions of an image resemble each other. Also there is self-similarity at every scale. A fully automated version of this technique was devised by Arnaud Jacquin, using partitioned IFS (PIFS) (1992)[4, 5], which is called Baseline Fractal Image Compression (BFIC)[4].

The most critical problem this technique faces is its slow compression step. A huge amount of research has been done to improve the performance of this technique which mainly includes:-

I. Better partitioning Scheme, II. Reducing the number of domain pool, III. Reducing number Of domain and range comparison or better classification, IV. Effective encoding scheme.

But the major drawback of the Jacquin Scheme where it lacks behind other image compression techniques like jpeg (DCT [8] based image compression) or wavelet based technique, is the large number of domain-range comparisons that goes into Fractal Image Compression. This is the most time consuming step.

2 Review

Plenty of improvements have been suggested by scientists since then, for a better classification scheme (i.e. to reduce the number of searches/comparisons). For example, Jacquin classified blocks on their edge content and Jacobs et al. used block brightness orien-
tion [4, 5]. Monro and Dudbridge [9] localized the domain pool with domain blocks closing to a given range block. Saupe [11] excluded the domain blocks with the smallest variance. Bani-Eqbal [12] arranged the domain block in a tree structure to direct the search and speed up the searching without noticeable loss of image quality. In this paper we proposed a fixed size partition of domain pool and so on range block using hierarchical classification[14] scheme to speed up FIC.

3 Fractal Image Compression (FIC)

3.1 Mathematics

The mathematical analogue of a partition copying machine is called a partition iterated system (PIFS)[13]. The definition of a PIFS is not dependent on the type of transformations, but in this paper we will use affine transformations. There are two spatial dimensions and the grey level adds a third dimension, so the transformations \( w_i \) are form:

\[
W_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_{i,1} & a_{i,2} & 0 \\ a_{i,3} & a_{i,4} & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} d_{i,1} \\ d_{i,2} \\ 0 \end{bmatrix} \tag{1}
\]

An affine transformation in \( R^3 \) is a function consisting of a linear transformation and translation in \( R^3 \). Affine transformations in \( R^2 \), for example, are of the form:

\[
W(x, y) = (ax + by + e, cx + dy + f) \tag{2}
\]

Where the parameters \( a, b, c, \) and \( d \) form the linear part, which determines the rotation, skew, and scaling; and the parameters \( e \) and \( f \) are the translation distances in the \( x \) and \( y \) directions, respectively.

Affine transformations in \( R^3 \):

**Definition 1** Let \( X \) be a metric space with metric \( d \). A map \( w : X \rightarrow X \) is contractive mapping with a contractivity factor \( s \) if there exists a positive real value \( s \) such that

\[
d(w(x), w(y)) \leq sd(x, y) \tag{3}
\]

for every \( x, y \in X \). If the contractivity constant satisfies \( s < 1 \), then \( w \) is said to be contractive with contractivity \( s \) [13].

For our purpose, only contractive affine transformations are considered. An affine map is said to be contractive if its contractivity factor \( s \) is less than 1, i.e. it “reduces” the size of the domain images.

**Definition 2** Let \( W = \{w_1, w_2, ..., w_m\} \) be an IFS with contractivity factor \( s, 0 \leq s < 1 \). Let \( W:H(R^n) \rightarrow H(R^n) \) denote its associated transform, and let \( A \in H(R^n) \) denote the attractor of \( W \). Then

\[
d(L, A) \leq \frac{d(L, W(L))}{1 - s} \tag{4}
\]

This theorem implies is that for a given compact subset \( L \in H(R^n) \) and for a given IFS \( W = \{w_1, w_2, ..., w_m\} \) such that \( d(L, W(L)) \) is very small, the attractor of \( W, A \in H(R^n) \), is very close to \( L \).

Given a finite collection of contractive affine maps \( w_1, w_2, ..., w_m \), this collection forms an iterated functions system (IFS). If \( B \) is a compact, nonempty subset of \( R^n \), then the map

\[
W(B) = \bigcup w_i(B) \tag{5}
\]

is a contractive map on \( H \), the (complete) metric space of compact sets in \( R^n \). \( W \) has a unique fixed point in \( H \), say \( A \). Then \( A \) is a compact subset of \( R^n \), and is called the attractor of \( W \).

A collage of \( B \) is an IFS \( W \) such that \( W(B) = \bigcup w_i(B) \) is approximately equal to \( B \). The Collage Theorem [2] states that if an IFS gives a collage of \( B \) which is a very good approximation of \( B \) (i.e. \( B \) and \( W(B) \) are very close in Hausdorff distance), then the attractor will be very close to (looks like) the original set \( B \).

3.2 Theory

The fractal image coding has the advantage of very fast decompression as well as the promise of potentially very high compression ratios. Another advantage of fractal image compression is its multi-resolution property, an image can be decoded at higher or lower resolutions than the original, and it is possible to “zoom-in” on sections of the image [10]. These advantages make fractal-image coding technique a very excellent method in multimedia applications: for example, Microsoft adopted it for compressing thousands of images in its Encarta multimedia encyclopedia [3].

Let we have the size of the original image \( f \) is \( M \times N \). During the process of fractal image coding, image \( f \) will be segmented into non-overlapping range blocks. The size of each range block is \( B \times B \). For each range block \( f_i \), we search the domain pool \( \Omega \) in order to get one domain block \( f_i : D_k \), which is similar with the range block in some sense and then apply contractive affine transformation eq.(2) \( W_k \) to \( f_i : D_k \). The optimal selection of \( f_i : D_k \) and \( W_k \) must assure
that the image \(g = \{W_k(f;D_k), k = 1,2,\ldots,N\}\), which is obtained by transforming \(f:D_k W_k\) \(k = 1,2,\ldots,N\), has the minimum difference from \(f = \{f:R_k, k = 1,2,\ldots,N\}\). The size of \(f:D_k\) must be larger than the size of \(f:R_k\) to assure \(W_k\) contractive see definition 1. Conceptually we set the size of domain block \(2B \times 2B\) from the top left corner of original image \(f\) with integer step (known as domain density) in horizontal and vertical directions. When domain density equals 1, the searching is complete searching and there will be \((M - 2 \times B +1) \times (M - 2 \times B +1)\) domain blocks in domain pool. 1.

The basic fractal image encoding scheme is given in the pseudo-code in Algorithm [17]

**Algorithm 1 Basic Fractal image encoding algorithm**

1: procedure BFIC
2: Loop:
3: Range Block \(\leftarrow\) for every range block \(R_i\),
4: i = 1,2,\ldots,N_R ,do
5: Loop:
6: Domain Search \(\leftarrow\) for every domain block \(D_j\),
7: j = 1,2,\ldots,N_D, do
8: Loop a:
9: For every \(a_k, k = 1,2,\ldots,m\), do
10: Loop b:
11: For every \(b_l, 1 = 1,2,\ldots,n\), do
12: Error Calculation
13: \[\text{error} = \| a_k D + b_l I - R \| ^2 \] (6)

Here, \(N_R\) is the number of range blocks, \(N_D\) is the number of domain blocks in domain pool \(\Omega\).

## 4 Problem Of Exhaustive Search

As describe in section 1, a very large number of domain-range comparison is the main difficult of the fractal encoding algorithm. Experiments on standard images, consider an image of size \(N \times N\). Let the entire image is partitioned into \(M \times M\) non-overlapping range blocks. The total number of range blocks are given by \(\frac{(N M)^2}{2}\). Most implementation use the size of domain block is twice larger than the range block i.e \(2 \times M\). Let the total number of domain blocks are given by \((N - 2M + 1)^2\). The domain blocks are overlapping. In Algorithm 1, there are nested LOOP in the process and for every step we need to calculate the error defined by eq. 6. The computation of best matching between a range block and a domain block is \(O(M^2)\). Considering \(M\) to be a constant, the computation complexity domain search for a range is \(O(N^4)\), which is approximately exponential time. Encoding time can be reduced by reducing the size of the domain pool \([15]\).

## 5 Fisher’s Classification Scheme

The domain-range comparison step of the image encoding is very computationally intensive. We use a classification scheme in order to reduce the number of domains blocks compared with a range blocks. The classification scheme is the most common approach for reducing the computational complexity. In such classification schemes, domain blocks are grouped into number of classes according to their common characteristics. For fractal image decoding, the decoding provides a very high quality image in a few iterations without any change in compression ratio \([16]\).

Fisher’s classification scheme \([13]\) is as follows: A square sub-image (domain or range) is divided into upper-left, upper-right, lower-left, and lower-right quadrants, numbered sequentially.

On each quadrant, values \(A_i\) (proportional to mean pixel intensity) and \(V_i\) (proportional to pixel intensity variance) are computed. If the pixel values in \(i^{th}\) quadrant are \(r_1^i, r_2^i, r_3^i, \ldots, r_n^i\) for \(i = 0,1,2,3\), we compute

\[A_i = \sum_{j=1}^{n} r_j^i\] (7)

and

\[V_i = \sum_{j=1}^{n} (r_j^i)^2 - A_i^2\] (8)

After that it is also possible to rotate the sub-image (domain or range) such that the \(A_i\) are ordered in one of the following three ways:
Major Class 1: $A_1 \geq A_2 \geq A_3 \geq A_4$
Major Class 2: $A_1 \geq A_2 \geq A_4 \geq A_3$
Major Class 3: $A_1 \geq A_4 \geq A_2 \geq A_3$

These orderings constitute three major classes and are called canonical orderings. Under each major class, there are 24 subclasses consisting of $^4 P_1$ orderings of variance ($V_i$) from eq. 8. Thus there are 72 classes in all. In this paper, we refer to this classification scheme as FISHER72.

According to the fisher that the distribution of domains across the 72 classes was far from uniform [13]. So fisher went on to further simplify the scheme. 24 classes in the FISHER72 classification. Fisher concluded: "the improvement attained by using 72 rather than 24 classes is minimal and comes at great expense of time" [13]. In this paper, we refer to this modified form of FISHER72 as FISHER24 using this concepts a hierarchical classification is proposed by N. Bhatcharya et al. [14]. We simply take the advantages of hierarchical classification [14] of sub-images and combining with fixed size partition to reduce the encoding time.

6 Proposed Technique

Fisher used values proportional to the mean and the variance of the pixel intensities to classify the domain and range image. In our proposed schemes Algorithm 2, we use only the sum of pixel intensities of fixed parts of domain (8 x 8) or range (4 x 4) then classify those fixed part.

Figure 2: Domain pool has domains with fixed size of 8 x 8 and 24 classes (child) from domain of size 8 x 8 in Level I. There are 331776 classes (child) for every 24 classes in Level I create Level II. Every nodes of Level II have 331776 array cells point to a list of domains together in that class.

According to the proposed a Algorithm 2 during compression, at first the domain pool is being created, data structures are defined as in the Figure 2. Domains are first classified by their size, then into Level-I, according to pixel-value sum of 4 quadrants, and finally into Level-II, according to pixel-value sum of 16 sub-quadrants. After two Levels of classification domain is place in list of point to array known as domain pool Figure 2.

Algorithm 2 A Speeding Up Fractal Image Compression using Fixed Size Partition and Hierarchical Classification of Sub-images

1: procedure Proposed
2: Range Pool (R) ← The image is partitioned into non-overlapping Fixed size range (4 x 4).
3: Domain Pool (D) ← The image is partitioned into overlapping Fixed size domain (8 x 8).
4: Loop ← Each range block is then divided into upper left, upper right, lower left and lower right each part is known as quadrant ($S_i$).
5: Thus we observe that there can be in total $^4 P_4$ (24) permutations possible, based on the relative ordering of the summation of pixel intensities and a corresponding class (class - 1 to 24) is assigned to it.
6: Each of the quadrant is further sub-divided into four sub-quadrants.
7: The sum of pixel values $S_{i,j}$ (i = 0,1,2,3; j = 0,1,2,3) for each sub-quadrant are calculated.
8: We again obtain the classes each of the sub-quadrants (class 1 to 24) i.e. for a particular a range /domain block we obtain 16 sub-quadrants or the domain pool can be classified into $24^4 = 331776$ classes.

In the proposed compression algorithm, when searching the domain pool for a best-match with a particular range, only those domains that are in the same Level-II and same class.

7 Results and Discussions

7.1 System Tools

Five standard 512 x 512 x 8 grayscale images have been used to test the proposed techniques 6 and also for comparison with FISHER24 classification scheme and modified Hierarchical classification [14].
The algorithm was implemented in C++ programming language running on a PC with following specifications: CPU Intel Core 2 Duo 2.0 GHz; RAM 4 GB; OS Ubuntu 14.4 64.

### 7.2 Research Result

The Comparison of compression time for the five image files have been made in Table 1. The comparison of PSNRs for the same image are given in Table 2 while space saving are given in Table 3 and decoding time of image are in Table 4. The pictorial representation of compression times, PSNRs, space savings and decoding times are illustrated in Figures 3, 4, 5 and 6 respectively. Figure 7 show the close up of Lenna original, decoded after using existing as well as proposed one.

#### Table 1: Comparison of encoding time(s) of Images

<table>
<thead>
<tr>
<th>Image data</th>
<th>BFIC</th>
<th>Paper [14]</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial</td>
<td>291.081</td>
<td>72.781</td>
<td>0.451</td>
</tr>
<tr>
<td>Baboon</td>
<td>304.790</td>
<td>84.618</td>
<td>0.437</td>
</tr>
<tr>
<td>Boat</td>
<td>309.488</td>
<td>85.425</td>
<td>0.439</td>
</tr>
<tr>
<td>Bridge</td>
<td>322.336</td>
<td>88.303</td>
<td>0.441</td>
</tr>
<tr>
<td>Lenna</td>
<td>283.244</td>
<td>72.949</td>
<td>0.492</td>
</tr>
</tbody>
</table>

#### Table 2: Comparison of PSNRs of Images(in dB)

<table>
<thead>
<tr>
<th>Image data</th>
<th>BFIC</th>
<th>Paper [14]</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial</td>
<td>38.67</td>
<td>26.32</td>
<td>23.74</td>
</tr>
<tr>
<td>Baboon</td>
<td>36.36</td>
<td>25.61</td>
<td>25.61</td>
</tr>
<tr>
<td>Boat</td>
<td>41.93</td>
<td>26.01</td>
<td>26.01</td>
</tr>
<tr>
<td>Bridge</td>
<td>39.46</td>
<td>27.43</td>
<td>25.62</td>
</tr>
<tr>
<td>Lenna</td>
<td>41.63</td>
<td>32.33</td>
<td>29.22</td>
</tr>
</tbody>
</table>

#### Table 3: Comparison of Space Savings (%) of Images

<table>
<thead>
<tr>
<th>Image data</th>
<th>BFIC</th>
<th>Paper [14]</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial</td>
<td>60.94</td>
<td>64.63</td>
<td>91.71</td>
</tr>
<tr>
<td>Baboon</td>
<td>53.80</td>
<td>59.36</td>
<td>92.07</td>
</tr>
<tr>
<td>Boat</td>
<td>56.76</td>
<td>57.27</td>
<td>90.43</td>
</tr>
<tr>
<td>Bridge</td>
<td>56.12</td>
<td>56.34</td>
<td>90.40</td>
</tr>
<tr>
<td>Lenna</td>
<td>64.03</td>
<td>64.23</td>
<td>90.23</td>
</tr>
</tbody>
</table>

#### Table 4: Comparison of Decoding time (s) of Images

<table>
<thead>
<tr>
<th>Image data</th>
<th>BFIC</th>
<th>Paper [14]</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial</td>
<td>0.133</td>
<td>0.125</td>
<td>0.094</td>
</tr>
<tr>
<td>Baboon</td>
<td>0.137</td>
<td>0.128</td>
<td>0.091</td>
</tr>
<tr>
<td>Boat</td>
<td>0.134</td>
<td>0.128</td>
<td>0.094</td>
</tr>
<tr>
<td>Bridge</td>
<td>0.136</td>
<td>0.133</td>
<td>0.095</td>
</tr>
<tr>
<td>Lenna</td>
<td>0.131</td>
<td>0.126</td>
<td>0.099</td>
</tr>
</tbody>
</table>

### 8 Conclusion

The proposed Fractal image encoding by using Fixed size partition and hierarchical classification of domain
and range improves the compression time of images significantly, when compared to existing FISHER24 classification as well as our Fractal image compression using hierarchical classification of sub-image and quadtree partition. PSNRs of decoded images using proposed scheme compared FISHER24 and other papers till date are approximately closer.

References:


