Fundamenta Informaticae 34 (1998) 1–16 1
IOS Press

A Study on Partitioned Iterative Function Systems for Image Compression

Suman K. Mitra, C. A. Murthy and Malay K. Kundu
Machine Intelligence Unit,
Indian Statistical Institute,
203, B. T. Road,
Calcutta 700035.
INDIA.
Email: res9432@isical.ac.in
murthy@isical.ac.in
malay@isical.ac.in

Abstract. The technique of image compression using Iterative Function System (IFS) is known as fractal image compression. An extension of IFS theory is called as Partitioned or local Iterative Function System (PIFS) for coding the gray level images. The theory of PIFS appears to be different from that of IFS in the sense of application domain. Assuming the theory of PIFS is same as that of IFS, several techniques of image compression have been developed. In the present article we have studied the PIFS scheme as a separate one and proposed a mathematical formulation for the existence of its attractor. Moreover the results of a Genetic Algorithm (GA) based PIFS technique [1] is presented. This technique appears to be efficient in the sense of computational cost.

1. Introduction

The theory of fractal based image compression using Iterative Function System (IFS) was proposed by Barnsley [2, 3]. He modeled real life images by means of deterministic fractal objects *i.e.*, by the attractor evolved through iterations of a set of contractive affine transformations. Once the set of contractive affine transformations $\mathcal{F}$ (say) is obtained the rest is an iterative sequence to produce the attractor which is an approximant of the given image. The set of contractive affine transformations $\mathcal{F}$ is called IFS. In particular at the Nth iteration, the object which is used as input to the IFS, is the output object obtained from the $(N-1)th$ iteration.

The detailed mathematical description of the IFS theory and other relevant results are available in [2, 3, 4, 5, 6].

Image compression using IFS can be looked upon as an inverse problem of iterative transformation theory [7]. The basic problem here is to find appropriate contractive affine transformations whose attractor is an approximation of the given image. Thus for the purpose of image compression it is enough to store the relevant parameters of the said transformations instead of the whole image. A fully automated fractal based image compression technique of digital monochrome image was first proposed by Jacquin [7, 8, 9]. This technique is known as partitioned [10] or local [3] iterative function system. The partitioned/local IFS is an IFS where the domain of application of the contractive affine transformations is restricted to the small portions of the image (subimages) instead of the whole image as in the case of IFS.

Different schemes, using PIFS, have been proposed by several other researchers [10, 11, 1, 12]. So far the efficiency, either in terms of quality of the image or in terms of computational cost, of the PIFS technique is the main interest of the researchers. But the theory of partitioned/local IFS appears to be different from the theory of IFS in the sense of restriction of the application domain for the contractive affine transformations. So, the questions are, how does PIFS produce an attractor. The present article provides a mathematical formulation for the existence of an attractor of partitioned IFS, assuming it as a separate scheme. To complete the study of PIFS for image compression, the results of a Genetic Algorithm (GA) based PIFS technique [1] have also been shown.

Genetic algorithms (GAs) [13, 14] are mathematically modeled algorithms which try to emulate biological evolutionary processes to solve optimization problems. Instead of searching one point at a time (usual technique adopted in enumerative search), GAs possess multiple search points. GAs attempt to find near optimal solutions without going through an exhaustive search mechanism. Thus GAs have an advantage of significantly large reduction in search space and time particularly when the search space is very large.

In the next section we have described the theory of image coding using IFS. Section 3 consists of basic features of constructing PIFS codes for a given image and the basic difference of PIFS with IFS. The proposed mathematical formulation of PIFS has been discussed in Section 4. The principles of GA and its use to find PIFS is discussed in Section 5. The experimental results have been presented in Section 6 and the conclusions are drawn in Section 7.

2. Basic Theory of IFS for Image Coding

The salient features of IFS theory and image coding through IFS are given below.

Let (X, d) be a metric space, where X is a set and d is a metric. Generally, X is taken as the collection of compact sets and d is taken as distance measure between two sets in X. Let f be a contractive affine map defined on metric space (X, d) such that $f : X \to X$ and $d(f(x_1), f(x_2)) \leq s\ d(x_1, x_2);\ \forall x_1, x_2 \in X$, where $0 \leq s < 1$ is called contractivity factor of the map f. For any large positive number N, $\lim_{N\to\infty} f^N(x) = a,\ \forall x \in X$, and also $f(a) = a$. "a" is

called fixed point (attractor) of f. Here $f^N(x)$ is defined as

$$f^N(x) \;=\; f(\; f^{N-1}(x)\;),\; \text{with } f^1(x) = f(x),\; \forall x \in X.$$

Now, let I be a given grayscale image which belongs to the set X. Our intention is to find a set $\mathcal{F}$ of affine contractive maps for which the given image I is an approximate fixed point. $\mathcal{F}$ is constructed in such away that the distance between the given image and the fixed point (attractor)of $\mathcal{F}$ is very small. To any set S belongs to X, the set of transformations $\mathcal{F}$ is used as follows;

$$\mathcal{F}(S) = \bigcup_i f_i(S).$$

The attractor "A" of the set of maps $\mathcal{F}$ is defined as follows :

$$\lim_{N\to\infty} \mathcal{F}^N(J) \;=\; A, \quad \forall J \in X,$$

and $\mathcal{F}(A) \;=\; A,$ where $\mathcal{F}^N(J)$ is defined as

$$\mathcal{F}^N(J) \;=\; \mathcal{F}(\; \mathcal{F}^{N-1}(J)\;), \text{with}$$

$$\mathcal{F}^1(J) \;=\; \mathcal{F}(J), \quad \forall J \in X.$$

Also the set of maps $\mathcal{F}$ is defined as follows;

$$d(\mathcal{F}(J_1), \mathcal{F}(J_2)) \leq s\; d(J_1, J_2);\; \forall J_1, J_2 \in X \quad \text{and} \quad 0 \leq s < 1. \tag{1}$$

s is called the contractivity factor of $\mathcal{F}$.

$$\text{Let} \;\; d(I, \mathcal{F}(I)) \leq \epsilon \tag{2}$$

where ϵ is a small positive quantity. Now, by Collage theorem [2], it can be shown that

$$d(I, A) \leq \frac{\epsilon}{1-s} \tag{3}$$

where A is the attractor of $\mathcal{F}$.
From (3) it is clear that, after a sufficiently large number(N) of iterations, the set of affine contractive maps $\mathcal{F}$ produces a set which belongs to X and is very close to the given original image I. Here, $(X, \mathcal{F})$ is called iterative function system and $\mathcal{F}$ is called the set of fractal codes for the given image I.

In the context of digital monochrome image, the coding scheme is called partitioned or local Iterative Function System. In the next section, the construction of PIFS codes has been described along with its basic difference from IFS.

3. Technique of PIFS

The structure of PIFS codes are almost same as that of IFS codes. The only difference is that PIFS codes are obtained and applied to a particular portion of the image instead of the whole image. The technique of construction of PIFS is given below.

3.1. Construction of PIFS

Let, I be a given grayscale image having size $w \times w$ and the range of gray level values be $[0, g]$. Thus the given image I is a subset three dimensional Euclidian space ($I \in \mathbb{R}^3$). The image is partitioned into n non overlapping squares of size, say $b \times b$, and let this partition be represented by $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_n\}$. Each $\mathcal{R}_i$ is named as range block. Note that $n = \frac{w}{b} \times \frac{w}{b}$ and $I = \bigcup_{i=1}^{n} \mathcal{R}_i$. Let $\mathcal{D}$ be the collection of all possible blocks which is of size $2b \times 2b$ and all are within the image support. Let $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_m\}$. Each $\mathcal{D}_j$ is named as domain block with $m = (w - 2b) \times (w - 2b)$ and $\mathcal{D}_j \subset [0, w] \times [0, w]$.

Let,

$$\mathcal{F}_j = \{f : f(\mathcal{D}_j) \to \mathbb{R}^3 \; ; \quad f \text{ is an affine contractive map}\}.$$

Now, for a given range block $\mathcal{R}_i$, let, d be distance measure and $f_{i|j} \in \mathcal{F}_j$ be such that

$$d(\mathcal{R}_i,\ f_{i|j}(\mathcal{D}_j)) \ \leq \ d(\mathcal{R}_i,\ f(\mathcal{D}_j)) \ \ \forall f \in \mathcal{F}_j,\ \forall j.$$

Now let k be such that

$$d(\mathcal{R}_i,\ f_{i|k}(\mathcal{D}_k)) \ = \ \min_j\{\ d(\mathcal{R}_i,\ f_{i|j}(\mathcal{D}_j))\ \} \tag{4}$$

Also, let $f_{i|k}(\mathcal{D}_k) = \widehat{\mathcal{R}}_{i|k}$.

Our aim is to find $f_{i|k}(\mathcal{D}_k)$ for each $i \in \{1, 2, \cdots, n\}$. In other words, for every range block $\mathcal{R}_i$, one needs to find an appropriately matched domain block $\mathcal{D}_k$ as well as an appropriate transformation $f_{i|k}$. The set of maps $\mathcal{F} = \{f_{1|\bullet}, f_{2|\bullet}, \cdots, f_{n|\bullet}\}$ thus obtained is called the partitioned or local IFS or fractal codes of image I.

To find the best matched domain block as well as the best matched transformation, we are to search all possible domain blocks as well as all possible transformations with the help of equation (4). The Problem of searching appropriately matched domain block and transformation for a range block can be solved by enumerative search [8] or by using Genetic Algorithms [1].

The affine contractive transformation $f_{i|\bullet}$ is constructed using the fact that the gray values of the range block are scaled and shifted version of the gray values of domain block. The contractive affine transformation $f_{i|j}$ defined on $\mathbb{R}^3$ is such that $f_{i|j}(\mathcal{D}_j) \to \mathcal{R}_i$. Also $f_{i|j}$ consists of two parts, one for spatial information and the other for information of gray values. The first part indicates which pixel of the range block corresponds to which pixel of domain block. The second part is to find the scaling and shift parameters for the set of pixels of the domain blocks to the range blocks.

The first part is shuffling the pixel values of the domain block and can be achieved by using any one of the eight possible transformations (isometry) on the domain blocks[8]. Once the first part is obtained, second part is estimation of a set of values (gray values) of range blocks from the set of values of the transformed domain blocks. These estimates can be obtained by using the least square analysis of two sets of values [1].

The second part is obtained using least square analysis of two sequences of gray values, one from the range block and other from the domain block, once the first part is fixed. Moreover the size of the domain blocks is double that of the range blocks. But, the least square (straight line fitting) needs point to point correspondence. To overcome this, one has to construct contracted domain blocks such that the number of pixels in the contracted domain blocks become equal to that of range blocks. The contracted domain blocks are obtained by adopting any one of the two techniques. In the first technique the average values of four neighboring pixel values in a domain blocks are considered as the pixel values of the contracted domain blocks [8]. In the other scheme, contracted domain blocks are constructed by taking pixel values from every alternative rows and columns of the domain blocks [1]. The proposed mathematical formulation 4 of PIFS is based on this second scheme.

Now to select appropriately matched domain block ($\mathcal{D}_k$) and appropriately matched transformation ($f_{i|k}$) for a range block ($\mathcal{R}_i$), the distance measure "d" plays an important role. The distance measure "d" [used in equation (4)] is taken to be the simple Mean Square Error (MSE) between the original set of gray values and the obtained set of gray values of the concerned range block. The MSE is not a metric though it serves the purpose of a distance measure. As selection of PIFS code for a range block is dependent only on the estimation of pixel values of that block, it is enough to calculate only the distortion of the original and estimated pixel values of the block. Note that the same measure had been used in all most all the articles [8, 10, 11, 1, 12].

3.2. How PIFS technique differs from IFS

An extension of the iterative function system concept is the partitioned iterative function system. PIFS mainly differs from IFS in the domain of application of their respective transformations. In PIFS the transformations are not applied to the whole image, as in the case of IFS, but rather have restricted domains. In all PIFS, a transformation f_i is specified not only by an affine map, but also by the domain to which f_i is applied.

In PIFS the map $f_{i|j}$ is applied to the domain $\mathcal{D}_j$ to result in $\widehat{\mathcal{R}_i}$, which is an estimate of $\mathcal{R}_i$. In the next iteration, this estimate ($\widehat{\mathcal{R}_i}$) is not used as the input to the map $f_{i|j}$. In particular in the next iteration an estimate of $\mathcal{D}_j$ is used as the input to obtain improved estimate, $\widehat{\widehat{\mathcal{R}_i}}$, of $\mathcal{R}_i$. A domain block may includes many other range blocks or part of them. So, the estimate of $\mathcal{D}_j$ consists of several other estimated range blocks or part of them.

Another important and significant difference of PIFS and IFS lies in the context of contractivity factor of the transformations. For an IFS with an expansive map f_i, the set of maps will not converge to a compact fixed point. The expansive part will cause the limit set to be infinitely stretched along some direction. This is not necessarily true for a PIFS. PIFS can contain expansive transformations and still have a bounded attractor. So, it is not necessary, in PIFS, to impose any contractivity condition on all the transformations. A necessary and sufficient contractivity requirement is that the set of transformations $\mathcal{F}$ be *eventually contractive* [10]. Fisher et al [10] have shown experimentally that maximum allowable value of s (contractivity factor) can be 1.5 (> 1). Also they have shown that this maximum value of s, for a particular

image, yields minimum distortion between the original image and the attractor evolved through the iterative process of the eventually contractive transformations.

In the next section we have described the mathematical formulation of attractor of PIFS.

4. Mathematical Formulation for the Existence of Attractor of PIFS

In this section we have proposed a mathematical formulation for the existence of attractor of PIFS. In particular we have shown that the PIFS codes $\mathcal{F}$ possesses a fixed point or attractor in an iterative sequence.

Let, I be a given image having size $w \times w$ and the range of gray level values be $[0, g]$. For this given image we can construct a vector $\underline{x}$ whose elements are the pixel values of the given image I. Note that there are w^2 pixel values of I. Thus,

$$\underline{x} \;=\; (x_1, \quad x_2, \quad x_3, \quad \ldots \quad , x_{w^2})'$$

is the given image where x_1 is the pixel value corresponding to the $(1,1)th$ position of I. Likewise, let x_r be the pixel value corresponding to the $(i,j)th$ position of I, where, $r \;=\; (i \;-\; 1)\; w \;+\; j, \quad 1 \;\leq\; i, \;\; j \;\leq\; w.$

In this setup PIFS can be viewed as following. There exists an affine (linear), not necessarily strictly contractive, map for each element of $\underline{x}$ and this map is called forward map of the element. In the process of iteration, the input to a forward map will be any one of the w^2 elements of $\underline{x}$ and the map is called backward map for this input element. Thus for each element of $\underline{x}$ there exists a forward map and an element of $\underline{x}$ can have one or more or no backward map(s). The set $\mathcal{F}$, of forward maps, is called the PIFS codes of I.

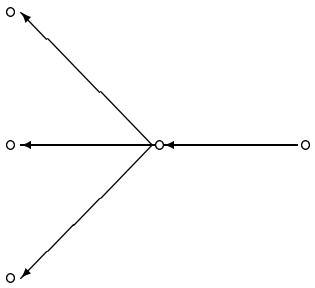

Fig. 1: Sketch of Forward map and backward maps

Now let us consider the set S where,

$$S \;=\; \{\; \underline{x} \;\mid\; \underline{x} \;=\; (x_1, \quad x_2, \quad x_3, \quad \ldots \quad , x_{w^2})' \;, \quad 0 \;\leq\; x_i \;\leq\; g \;\}.$$

S is the set of all possible images. The given image I is surely an element of S *i.e.* $I \in S$. The PIFS codes $\mathcal{F}$ can be looked upon as

$$\mathcal{F} : S \rightarrow S .$$

The attractor of $\mathcal{F}$, $\underline{a}$ (say), if exists will also belongs to S. So, the first task is to show the existence of $\underline{a}$.

Let f_1 be the forward map for a particular element x_{r_1}, where $r_1 = (i_1 - 1)w + j_1$. Also let this element be mapped from the element x_{r_2} ($r_2 = (i_2 - 1)\, w + j_2$). Thus f_1 is the backward map for xr_2. Again x_{r_2} is being mapped from x_{r_3}, $(r_3 = (i_3 - 1)w + j_3)$ with a forward map f_2. Thus we have a sequence of maps for the element x_{r_1} as following.

$$(i_1,\ j_1) \overset{f_1}{\leftarrow} (i_2,\ j_2) \overset{f_2}{\leftarrow} (i_3,\ j_3) \overset{f_3}{\leftarrow} \cdots \overset{f_{m-1}}{\leftarrow} (i_m,\ j_m);\quad m \leq (w^2 - 1) . \quad (5)$$

The above sequence will be stopped at $(i_m,\ j_m)$ if

$$(i_{m+1},\ j_{m+1}) = (i_k,\ j_k);\quad \text{for } k = 0 \text{ or } 1 \text{ or } 2 \text{ or } \cdots \text{ or } m. \quad (6)$$

The stopping phenomenon of this sequence is mandatory as there are finite number (w^2) of elements in $\underline{x}$. Moreover all the elements of $\underline{x}$ possess same type of sequence in PIFS codes. Thus it is enough to show that the element x_{r_1} has got a fixed point in the process of iteration and this will lead to prove the existence of $\underline{a}$ (attractor of $\underline{x}$).

It is clear from the sequence (5) that during the iterative process the element x_{r_1} will have a fixed point once the element x_{r_2} is fixed. Again the convergence (to a fixed point) of the element x_{r_3} confirms the convergence of the element x_{r_2} and likewise for the rest of the elements. Thus convergence of the last element of the sequence implies the convergence of the rest of the elements. The convergence of the last element of the sequence is possible in four different ways according to the stopping condition (6).

An important point to be noted in this context is the problem of discretization. To get the decoded image in an iterative process using PIFS codes one need to discretize the output. This can be done in two ways. One is discretization of the output in each iteration. Another is discretization at the end of the iterative process. The iterative process is stopped whenever there is no change in gray values in two successive iterations. To prove the convergence of the elements in four different ways we have used the discretization of the second type.

Case 1 : $m = 1$.
Here $(i_2,\ j_2) = (i_1,\ j_1)$.
It implies that $(i_1,\ j_1)$ is mapped into itself with a map f_1 .
Here $f_1 = a_1\, x + b_1$; $0 \leq x \leq g$ and $0 \leq a_1 < 1$.
Note that in this case the affine map f_1 should necessarily be a strictly contractive map otherwise the element will not converge to a fixed point.

If we start with any value ($0 \leq x \leq g$) of $(i_1,\ j_1)$, the element will converge to the fixed point $\frac{b_1}{1 - a_1}$.

Case 2 : $m > 0$ and $k = m$

Here $(i_{m+1}, j_{m+1}) = (i_m, j_m)$.

It implies :1 that (i_m, j_m) is mapped into itself with a map $f_m = a_m x + b_m$; $0 \leq x \leq g$ and $0 \leq a_m < 1$. Thus (i_m, j_m) will converge to $\frac{b_m}{1-a_m}$. Therefore the element (i_{m-1}, j_{m-1}) will converge to

$$\frac{a_{m-1}\, b_m}{1 - a_m} + b_{m-1} = \frac{a_{m-1}\, b_m - a_m\, b_{m-1} + b_{m-1}}{1 - a_m}.$$

In this case the forward map is $f_{m-1} = a_{m-1}\, x + b_{m-1}$; $0 \leq x \leq g$. Again (i_{m-1}, j_{m-1}) is fixed implies convergence of (i_{m-2}, j_{m-2}) with forward map $f_{m-2} = a_{m-2}\, x + b_{m-2}$; $0 \leq x \leq g$, at

$$\frac{a_{m-2}\, a_{m-1}\, b_m - a_{m-2}\, a_m\, b_{m-1} + a_{m-2}\, b_{m-1} - a_m\, b_{m-2} + b_{m-2}}{1 - a_m}.$$

Proceeding in this way, the fixed point of (i_1, j_1) is found out to be

$$\frac{a_1\, a_2 \cdots a_{m-1}\, b_m + (a_1\, a_2 \cdots a_{m-2}\, b_{m-1} + a_1\, a_2 \cdots a_{m-3}\, b_{m-2} + \cdots + a_1\, a_2\, b_3 + a_1\, b_2 + b_1)(1 - a_m)}{1 - a_m}.$$

Note that in this case the affine map f_m should necessarily be contractive. But the rest of the maps may be non contractive. The eventual contractivity, associated with the element $x_{r_1} = (i_1\ ,\ j_1)$, will be $s_{r_1} = \prod_{i=1}^{m} a_i$.

Case 3 : $m > 0$ and $k = 1$

Here $(i_{m+1}, j_{m+1}) = (i_1, j_1)$.

It implies that the starting and the last element of the sequence (5) is same. This can be looked as a complete loop for the sequence. This case has been solved stepwise. First of all the case is solved for $m = 2$, and $m = 3$. Then on the basis of these the fixed point for the case of general m is solved.

Case 3(a) : $m = 2$

Here we have only two elements viz. (i_1, j_1) and (i_2, j_2). The element (i_1, j_1) is being mapped from the element (i_2, j_2) by the affine map $f_1 = a_1 x + b_1$; $0 \leq x \leq g$. On the other hand the element (i_2, j_2) is being mapped from (i_1, j_1) by the affine map $f_2 = a_2 x + b_2$; $0 \leq x \leq g$.

$$(i_1, j_1) \overset{f_1}{\leftarrow} (i_2, j_2) \overset{f_2}{\leftarrow} (i_1, j_1).$$

Let x be the starting value of (i_1, j_1) and y be the starting value of (i_2, j_2). After first iteration the values of (i_1, j_1) and (i_2, j_2) will be $a_1\, y + b_1$ and $a_2\, x + b_2$ respectively. Again after second iteration these will be $a_1\, a_2\, x + a_1\, b_2 + b_1$ and $a_2\, a_1\, y + a_2\, b_1 + b_2$ respectively. Proceeding this way after infinite (practically large but finite) number of iteration, the fixed point of $(i_1\ ,\ j_1)$ and $(i_2\ ,\ j_2)$ will be independent of x and y. The Coefficients of x and y after N (even) iterations will be $(a_1\ a_2)^{N/2}$ which tends to zero as N tends to infinity. The fixed points of (i_1, j_1) thus will be

$$\frac{a_1\, b_2 + b_1}{1 - a_1\, a_2}.$$

The same for the element $(i_2,\ j_2)$ will be

$$\frac{a_2\ b_1\ +\ b_2}{1\ -\ a_1\ a_2}.$$

Note that both the maps need not be contractive. Moreover the eventual contractivity associated with the element x_{r_1} is $(a_1\ a_2)$ which should be less than one.

Case 3(b) : $m\ =\ 3$

Here we have three elements viz. $(i_1,\ j_1)$, $(i_2,\ j_2)$ and $(i_3,\ j_3)$. These three elements are making a complete loop in the sequence. The sequence of forward and backward maps is as follows;

$$(i_1,\ j_1)\ \overset{f_1}{\leftarrow}\ (i_2,\ j_2)\ \overset{f_2}{\leftarrow}\ (i_3,\ j_3)\ \overset{f_3}{\leftarrow}\ (i_1,\ j_1).$$

Taking the starting values of three elements as x, y and z and proceeding as *case 3(a)* we have the following results.

The fixed point of $(i_1,\ j_1)$ will be

$$\frac{a_1\ (\ a_2\ b_3\ +\ b_2\ )\ +\ b_1}{1\ -\ a_1\ a_2\ a_3}.$$

The element $(i_2,\ j_2)$ will converge to

$$\frac{a_2\ (\ a_3\ b_1\ +\ b_3\ )\ +\ b_2}{1\ -\ a_1\ a_2\ a_3}.$$

The fixed point of $(i_3,\ j_3)$ will be

$$\frac{a_3\ (\ a_1\ b_2\ +\ b_1\ )\ +\ b_3}{1\ -\ a_1\ a_2\ a_3}.$$

Here also the maps need not be contractive in the strict sense. The eventual contractivity will be $(a_1\ a_2\ a_3)$ in this case.

Case 3(c) : General m

Here we have m elements which are making a complete loop of sequence. It is clear from case *3(a)* and *case 3(b)* that all the elements of this sequence will have a fixed point after a large but finite number of iteration. Also the affine maps which are used, need not to be contractive. In particular, in this case the element $(i_1,\ j_1)$ will converge to

$$\frac{a_1\ (a_2\ (\ \ldots\ (a_{m-2}\ (a_{m-1}\ b_m\ +\ b_{m-1})\ +\ b_{m-2})\ +\ \ldots\ )\ +\ b_2)\ +\ b_1}{1\ -\ a_1\ a_2\ \ldots\ a_m}.$$

Also the eventual contractivity for the element is $s_{r_1}\ =\ \prod_{i=1}^{m} a_i$.

Case 4 : $m\ >\ 0$ and $0\ <\ k\ <\ m$

Here $(i_{m+1},\ j_{m+1})\ =\ (i_k,\ j_k)$, where $k\ =\ 2$ or 3 or $\ldots$ or $m-2$.

Without loss of generality say, $1\ <\ k\ =\ m_0\ <\ m-1$.

This case can be viewed as mixture two cases. Taking $(i_{m_0},\ j_{m_0})$ as the starting element,

a complete loop of sequence can be formed with rest of the elements. Thus, one can find the fixed point of this element as it is nothing but *case 3*. Once the element (i_{m_0}, j_{m_0}) is fixed then the fixed point of the original starting element (i_1, j_1) can be found out by using *case 2*. Like all the previous cases the eventual contractivity, in this case, will be $s_{r_1} = \prod_{i=1}^{m} a_i$.

Note that for each element there will be a sequence of the form (5). This sequence will follow any one of the above mentioned four cases. Thus for each element there will be a sequence of forward maps. The contractivity factor associated with this element will be the product of all the scaled parameters of the forward maps ($s_{r_1} = \prod_{i=1}^{m} a_i$). s_{r_1} defined here actually is Lipshitz coefficient. It becomes contractive coefficient if it is less than unity. Thus assuming the contractivity (eventual contractivity) we get $\mathcal{F}^N(\underline{o}) \rightarrow \underline{a} \ \forall \ \underline{o} \in S$ for a very large positive number N.

The next section deals with the GA based PIFS technique.

5. An efficient GA based technique of PIFS

For the better understanding of the GA based PIFS technique, we have furnished here a short description of GA.

5.1. Genetic Algorithms

Genetic Algorithms (GAs) are highly adaptive search and machine learning processes based on a natural selection mechanism of biological genetic system. GAs help to find the global near optimal solution without getting stuck at local optima as it deals with multiple points (called, chromosomes) simultaneously. To solve the optimization problem, GAs start with the chromosomal (structural) representation of a parameter set. The parameter set is coded as a string of finite length called a chromosome or simply a string. Usually, the chromosomes are strings of 0's and 1's. If the length of a string is l then total number of strings is 2^l.

Usually, a function "fit" is defined on the set of strings which represents the function to be optimized. This function "fit", also known as the "fitness function" denotes the fitness value of a string. In GAs, a string which provides optimal fitness value is found without exhaustive search. To find a near optimal solution, three basic genetic operators, i) Selection, ii) Crossover and iii) Mutation are exploited in GAs.

Out of all possible 2^l strings, initially a few strings [say S number of strings] are selected randomly and this set of strings is called initial population. Starting with the initial population the three genetic operators are used one by one to form a new population. The process of creating new population is called an iteration and is executed for a fixed number of times. Here we have also used the elitist model of GAs. In the elitist model of GAs the worst string in the present population is replaced by the best string of the previous population in each iteration to keep track with the best string obtained in each iteration.

In the selection procedure, S number of strings are selected from current population to form a mating pool. The selection of each individual string, as a string in the mating pool, is directly or inversely proportional to its fitness function as the problem is either a maximization or a minimization problem respectively. This type of selection scheme is known as proportional selection strategy. The crossover operation is then applied on the mating pool.

The most commonly used crossover operation is a single point crossover [13] operation on a pair of strings which is described here. An integer position k is selected randomly between 1 and $l-1$ ($l > 1$), l being the string length. Two new strings are then created by swapping all the characters from position $k+1$ to l of the old strings. The occurrence of crossover operation on a pair of strings is guided by crossover probability, say, P_{cross}. A random number (≤ 1) is drawn for each pair of strings. The drawn random number less than P_{cross} indicates the occurrence of crossover for that pair and the non occurrence if the reverse is true. Usually a high value is assigned for the crossover probability. The mutation operation is then applied.

In mutation operation every bit of every string is replaced by the reverse character (i.e. 0 by 1 and 1 by 0) with some probability. One of the commonly used conventions [13] is to assign a very small value to the mutation probability P_{mut} and keep the P_{mut} fixed for all the iterations.

The process is executed for a fixed number of times (iterations) and the best string, obtained so far, is taken to be the near optimal one. In the present article, the number of iterations, say T, is fixed a priori for the termination of GAs.

Description of the algorithm :- The genetic algorithm is implemented using the following steps.

1. Generate an initial population Q of size S and calculate fitness values of all S strings of Q.
2. Find the best string S_{bst} of Q. If the best string is not unique, then call any one of the best strings of Q as S_{bst}.
3. Construct a mating pool using proportional selection strategy (S_{bst} belongs to Q). Perform crossover and mutation operations on the strings in the mating pool and obtain a population Q_{tmp}. (If selection is ignored at the first iteration, it hardly makes any difference.)
4. Calculate the fitness value of each string S of Q_{tmp} and replace the worst string of Q_{tmp} by S_{bst}. Rename Q_{tmp} as Q.
5. If T iterations are completed then stop, else go to step 2.

Note that steps 2, 3 and 4 together make an iteration.

5.2. GA to Find PIFS

The main aspect of fractal based image coding is to find a suitable domain block and a transformation for a rough type range block. Thus the whole problem can be looked upon as a search problem. Instead of a exhaustive search mechanism GAs can be used to find the near optimal solution.

The number of possible domain blocks to be searched are $(w-2b) \times (w-2b)$ (section 3.1) The number of transformations to be searched for each domain block is 8 (section 3.1). Thus the space to be searched consists of M elements. M is called cardinality of the search space. Here $M = 8\ (w-2b)^2$. Let the space to be searched be represented by $\mathcal{P}$ where

$$\mathcal{P} = \{1, 2, \cdots, (w-2b)\} \times \{1, 2, \cdots, (w-2b)\} \times \{1, 2, \cdots, 8\}.$$

Binary strings are introduced to represent the elements of $\mathcal{P}$. The set of 2^l binary strings, each of length l, are constructed in such a way that the set exhausts the whole parametric space. In case of $2^l > M$, the strings which are outside the parameter space are ignored. To replace those strings, new strings are selected randomly. The value for l depends on the values of w and b. The fitness value of a string is taken to be the MSE between the given range block and the obtained range block.

Let S be the population size and T be the maximum number of iterations for the GA. Initially, S strings are selected randomly from 2^l strings, to result in an initial population for GA. The various steps of the GA, as mentioned in section 5.1 are implemented repeatedly up to T iterations. Note that the total number of strings searched up to T iterations is $S \times T$. Hence, $\frac{M}{S\ T}$ provides the search space reduction ratio for each rough type range block, and $(M - S\ T)$ provides the reduction in the search space for each rough type range block. If the number of smooth type range blocks is r then $(n-r) \times (M - (S\ T))$ would provide the total search space reduced for finding the set of transformations $\mathcal{F}$ for fractal image compression.

In the next section the experimental results of PIFS using exhaustive search mechanism and GA based technique have been presented.

6. Results

The GA based method and exhaustive search method are implemented in 256×256, 8 bit/pixel "Lena" image and "Seagull". The image is subdivided into four, 128×128 subimages, each of which is encoded separately. Fractal operators (codes) automatically take care of borders between subimages.

A simple classification technique [1] and a two level partition scheme [8] for the range blocks are used for the specific implementation of both techniques.In particular the range blocks are classified in to two classes one is called class of smooth blocks and the other is called class of rough blocks. In two level scheme parent range blocks of size 8×8 and children range blocks of size 4×4 are used [8].

Exhaustive search (ES) and GA are implemented, as a search technique, only for rough type range blocks. Here for each subimage, total number of parent range blocks is $n = 256$ and total number of domain blocks (m) to search is $(128-16) \times (128-16) = 112 \times 112$ and $(128-8) \times (128-8) = 120 \times 120$ for parent and child range blocks respectively. Thus the cardinality (M) of the search spaces for these two cases are $112 \times 112 \times 8$ and $120 \times 120 \times 8$ respectively. The string length l has been taken to be $17(7+7+3)$ in both the cases. As a

Table 1 Results obtained by using the GA based technique and the Exhaustive search technique for "Lena" image

Image	Genetic Algorithm			Exhaustive Search		
	CR	PSNR in db	Number of domain block searched	CR	PSNR in db	Number of domain block searched
Lena	10.50	30.22	8102640	11.24	28.32	161869952
Seagull	7.40	27.27	8102640	7.60	28.82	161869952

CR=Compression Ratio

PSNR=Peak Signal to Noise Ratio

result of selecting 2^{17} binary strings, a few strings, in both the cases, will be outside the specified search spaces. If a string which is not in the search space is selected during the implementation of the GA, then a string at the boundary will replace it.

(a) (b) (c)

Fig. 2: Original "Lena" image (a) with decoded "Lena" obtained from GA based method (b) and ES based method (c).

Out of these 2^{17} binary strings, 6 strings ($S = 6$) are selected randomly to construct an initial population. The total number of iterations considered in the GA is $T = 910$. Hence the search space reduction ratios for a parent and a child rough type range blocks are approximately 18 and 21 respectively. Test results for both images using both scheme are given in Table 1.

It is clear from Table 1 that more compression is achieved in exhaustive search. In this encoding scheme, in the first level, more rough type parent range blocks are encoded correctly. It implies that the parent rough type range blocks are not divided into child blocks for second level encoding. As a result of this more compression is achieved in exhaustive search case. But

the PSNR value appears to be better in case of GA based technique. Moreover the advantage of using GA based technique is established from the value of the number of domain blocks searched in both the cases. Here also GA based technique searched at least 20 times lesser number of domain blocks in comparison with the exhaustive search technique.

(a) (b) (c)

Fig. 3: Original "Seagull" image (a) with decoded "Seagull" obtained from GA based method (b) and ES based method (c).

The corresponding diagrams for the "Lena" image are shown in figure 2. Figure 2(a) shows the original "Lena" image, 2(b) and 2(c) are respectively decoded "Lena" where the codes are obtained by using exhaustive search and GA. On the other hand figure 3(a) is the original "Seagull" while 3(b) and 3(c) are decoded "Seagull" using exhaustive search and GA respectively. All the decoded images are obtained after ten iterations and starting from an arbitrary blank image.

7. Conclusions and further study

There are several techniques, available in the literature, for image compression using PIFS and in almost all the techniques it has been assumed that the theory of PIFS is same as that of IFS. Almost no emphasis has been given by the researchers for the modification or the development of the theory. The theory of IFS stands on a sound mathematical platform but the theory of PIFS appears to be different and a mathematical foundation of PIFS is needed. With this aim in mind we have proposed a mathematical framework to show the existence of the attractor of PIFS assuming it as a separate scheme. In particular we have sketched a detail description of the fractal operators (maps).

The PIFS based image compression techniques have their own advantages and disadvantages in the sense of compression ratio, peak-signal-noise-ratio and computational cost. But the comparison of these techniques with other image compression techniques specially with classical compression method (JPEG) or with Wavelet based image compression method is appreciated.

In this regard one can claim that the uniqueness of fractal image compression is that the reconstructed image can be obtained from any arbitrary image which is not the case of any other compression method. But at the same time one should really investigate the amount of loss incorporating in the compression ratio with such a strict constraint.

In PIFS technique the estimates of all the range blocks are obtained assuming the self similarities present in the given image. The scaled and transformed version of the domain block which is most similar to a range block is named as appropriately matched domain block for that range block. The similarity between the range block and the domain block is measured by MSE. Thus the efficiency of PIFS technique depends on two factors. The first one is the efficiency of the distortion measure and second one is the extent of similarity present in the domain blocks corresponding to the range blocks of a given image.

MSE being a global measure has its own limitations. In this context one can think of a better and reliable measure which can make the PIFS technique more efficient. Regarding the second factor, it may so happen that there is hardly any domain block which is appropriately matched with the concerned range block. In other words, the domain block most close to a range block in the sense of similarity, may provide a quantitatively large distortion measure. This may lead to inefficient coding. This problem can be viewed as a limitation of the PIFS based image compression technique. Thus, there is enormous scope for suggesting the specific theory which can make domain blocks more close to the range block in the sense of similarity and also with less distortion.

Acknowledgement

The first author is very glad to acknowledge **CRIT2 grant** for providing partial support during the preparation of the manuscript.

References

[1] S. K. Mitra, C. A. Murthy, and M. K. Kundu, "Technique for fractal image compression using genetic algorithm." *IEEE Transactions on Image Processing* (In Press).

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

[3] M. F. Barnsley and L. P. Hurd, *Fractal Image Compression*. Massachusetts: AK Press, 1993.

[4] J. Feder, *Fractals*. New York: Plenum Press, 1988.

[5] G. A. Edger, *Measure, Topology, and Fractal Geometry*. New York: Springer Verlag, 1990.

[6] K. Falconer, *Fractals Geometry Mathematical Foundations and Applications*. New York: John Wiley, 1990.

[7] A. E. Jacquin, *Fractal Theory of Iterated Markov Operators With Applications to Digital Image Coding*. PhD thesis, Georgia Institute of Technology, August 1989.

[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, pp. 18–30, 1992.

[9] A. E. Jacquin, "Fractal image coding : A review," *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1451–1465, 1993.

[10] Y. Fisher, E. W. Jacbos, and R. D. Boss, "Fractal image compression using iterated transforms," in *Image and Text Compression* (J. A.Storer, ed.), pp. 36–61, Kluwer Academic Publishers, 1992.

[11] S. K. Mitra, C. A. Murthy, and M. K. Kundu, "Fractal based image coding using genetic algorithm," in *Pattern Recognition, Image Processing and Computer Vision. Recent Advances* (P. P. Das and B. N. Chatterji, eds.), pp. 86–91, Narosa Publishing House, New Delhi, 1995.

[12] L. Thomas and F. Deravi, "Region-based fractal image compression using heuristic search," *IEEE Transactions on Image Processing*, vol. 4, no. 6, pp. 832–838, 1995.

[13] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning.* Reading, Massachussets: Addison - Wesley, 1989.

[14] L. Davis, *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold, 1991.