Image Compression and Edge Extraction Using Fractal Technique and Genetic Algorithm

Suman K. Mitra, C. A. Murthy, and M. K. Kundu

Machine Intelligence Unit
Indian Statistical Institute
203, B. T. Road, Calcutta 700035, INDIA.
E-mail: {res9432,murthy,malay}@isical.ac.in

Abstract. A large volume of image data is usually stored in the digital library system in a compressed form. Attempts are now being made to perform image processing tasks using the compressed form of images, which can be accessed directly from digital library. The present chapter is focused on two algorithms. In the first algorithm a fractal based image compression technique using genetic algorithms has been suggested. In particular, the genetic algorithm is used as a search technique to make present algorithm faster than the conventional fractal based image compression techniques. In the second one, a new method for extracting edges from the compressed image information has been described. Fractal code obtained from the first algorithm has been used as the input to the second algorithm. Thus the second one can be looked upon as an operation in the compressed domain. Actually, the process of extracting edges is embedded in the process of fractal reconstruction of the original image from the fractal code. Along with the reconstructed image, an edge image is obtained as a by-product. The scheme is unique of its kind as it is not using any kind of convolution operation based on kernels, which is very common in conventional edge detection schemes.

Key Words : Image Compression, Genetic Algorithm (GA), Edge Detection, Compression Ratio, Iterative Function System (IFS), Partitioned Iterative Function System (PIFS).

1 Introduction

The storage of a large volume of data, such as images in a digital library, usually requires huge memory space which necessitate some form of data compression. Moreover, it is becoming increasingly important to use the compressed form directly from the digital library instead of raw digital image for performing different image processing tasks required for various applications. With this aim in mind, we have, in this chapter, proposed a compression scheme using both fractal technique and genetic algorithms. A new method of edge extraction from the compressed image has also been described. Actually, edges are extracted from the intermediate reconstruction of the original image from the fractal codes. Hence the edge extraction process can also be looked upon as the by-product of the fractal image compression scheme.

S. K. Pal et al. (eds.), *Soft Computing for Image Processing*

The theory of image coding using iterative function system (IFS) was first proposed by Barnsley [1]. He modeled real life images by means of deterministic fractal objects i.e., by the attractors evolved through iterations of a set of contractive affine transformations. With the help of iterated function system along with Collage theorem, Barnsley laid the foundation stone of fractal based image compression. A set of contractive affine transformations can approximate a real image and so, it is enough to store the relevant parameters of the set of transformations requiring less memory.

The basis of fractal based image compression is quite different from conventional image compression schemes. The literature on fractal image compression has focused on two basic problems. The first problem is to determine a family of contraction maps that can be used to effectively code images [2]. The second problem is to find fast and effective algorithms for finding a set of contraction maps of which the given image is an approximate fixed point [3]. A review on fractal based image coding methodology is available in literature [4]. A fully automated fractal based image compression technique for digital image was first described by Jacquin [5]. In his work, Jacquin partitioned a given image into square blocks called range blocks. The encoding process consists of approximating these blocks from other portions of the image, called domain blocks, through some operations. As a result of this encoding process, separate transformations for each range block are obtained. The set consisting of these transformations, when iterated upon any initial image, will produce a fixed point (attractor) which approximates the target image. This scheme can be viewed as partitioned iterative function system (PIFS). Several algorithms, with different motivations, have been suggested to obtain PIFS or fractal code of a given image [6–8]. But the main disadvantage of all these techniques is mainly involved in encoding as usually they require a huge computational time. We have developed here a faster algorithm, to obtain PIFS, using genetic algorithm [9,10]. The proposed algorithm is faster than the technique described by Jacquin [5]. Actually, the effectiveness of the proposed algorithm in the sense of computational time depends on genetic parameters such as string length, initial population size and the termination condition. The detail description of genetic parameters are given in Section 3.1.

Genetic algorithms (GAs) [11–13] 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 use multiple search points. They attempt to find near optimal solutions without going through an exhaustive search mechanism. Thus in effect, GAs have an advantage of achieving a large reduction both in search space and time, particularly when the search space is very large. Genetic algorithms with elitist model are used in finding the appropriate domain block as well as the appropriate transformation for a range block.

This chapter also presents an edge extraction algorithm which is embedded in the image reconstruction sequence from the fractal code.

Edge extraction plays a very important role in many image processing applications. Usually, edges in an image are formed due to changes or discontinuities in image intensity. Hence one can say that edge points represent some features of an image. In the process of edge extraction, an edge image is produced from a gray level image. A great deal of effort has been directed towards finding sloution to this problem [14,15], but considerable success is yet to be achieved as edge semantics are extremely complicated. Our understanding of proper edge is still inadequate as there is hardly any measure to judge the true edge in an image. Most of the common algorithms for finding edges in an image use kernels known as edge detectors. These kernels are based on local gradients typically computed over a small window which scan the whole image. Now, from the gradient information, edges are computed using thresholding approaches [16–19].

Digital images used for modern image processing and analysis are usually of the form of a two dimensional array of pixels. Commonly, the images are accessed for analysis from the data base where images are stored in compressed form. The storage of a large volume of data in applications like remote sensing imagery and moving video image sequence require great deal of compression. The transmission of huge data through narrow band width channel also requires image compression. Therefore, instead of analyzing the raw digital images if one can develop methods for analyzing the coded representation of the image then advantages of such approaches will be of two fold. First it will reduce the storage requirements. Secondly it may appear to be cost effective in the sense of computational cost as no additional reconstruction of the image is needed in different image processing tasks when required. Similarly, in the case of transmission, if the coded version of the image is directly utilized at the receiving end for subsequent analysis then considerable amount of computer time is saved. So far only a few attempts have been made to utilize the coded version of the images for performing some specific image processing tasks [20,21]. With this problem in mind, an attempt is made to develop a new edge extraction technique which can be applied directly on the coded version of the image.

Theory and key features of IFS, image coding and decoding using PIFS and GAs are outlined in Section 2. Some basic features of Genetic Algorithms and the methodology of finding fractal code using GAs is described in Section 3. In Section 4, the detail methodology of extracting edges using fractal code is described. Section 5 presents implementation and results. Discussion and conclusions are provided in Section 6.

2 Basic Theory of Fractal Image Compression

The detailed mathematical description of the IFS theory, Collage theorem and other relevant results are available in [1,2,22]. Only the salient features are discussed here.

2.1 Theoretical Foundation of IFS

Let I be a given image which belongs to the set X. Generally X is taken as the collection of compact sets. Hence I can be looked upon as a set belonging to X. Our intention is to find a set $\mathcal{F}$ of affine contractive maps such that its fixed point will be a close approximation of the given image I. The fixed point or 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:

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

Here "d" is called the distance measure and "s" is called the contractivity factor of $\mathcal{F}$. Let

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

where ϵ is a small positive quantity. Now, by Collage theorem [1], it can be shown that

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

where "A" is the attractor of $\mathcal{F}$.

From equation (3) it is clear that, after a sufficiently large number of iterations, the set of affine contractive maps $\mathcal{F}$ produces a set which 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 code for the given image I.

2.2 Image Coding Using 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.

Let I be a given digital image having size $w \times w$ and the set of gray levels be $\{0, 1, 2, \cdots, l-1\}$. Thus the given image I can be expressed as a matrix

$((g(i,j)))_{w\times w}$, where i and j stand for row number and column number respectively and $g(i,j)$ represents the gray level value for the position (i,j). The image is partitioned into n non overlapping squares of size, say $b \times b$, and let this partition be represented by $\mathcal{N} = \{\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_n\}$. Each $\mathcal{R}_i$ is named as range block. Note that the number of range blocks is $n = \frac{w}{b} \times \frac{w}{b}$. Let $\mathcal{M}$ be the collection of all possible blocks of size $2b \times 2b$ within the image. Let $\mathcal{M} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_m\}$. Here $m = (w - 2b) \times (w - 2b)$ and $\mathcal{D}_j$'s are named as "domain blocks".

Now, let us define,

$$\mathcal{A} = \{1, 2, \cdots, w\} \times \{1, 2, \cdots, w\} \times \{0, 1, 2, \cdots, l-1\}.$$

Here $\mathcal{A} \subset \mathbb{R}^3$. Note that any image I is a subset of $\mathcal{A}$ but any subset of $\mathcal{A}$ is not necessarily an image. Also $\mathcal{R}_i \subset \mathcal{A}; \forall i$ and $\mathcal{D}_j \subset \mathcal{A}; \forall j$.

Let, for a range block $\mathcal{R}_i$,

$$\mathcal{F}_j = \{f : \mathcal{D}_j \to \mathcal{A} \ ; \ f \text{ is an affine contractive map}\}.$$

Let, $f_{i|j} \in \mathcal{F}_j$ be such that

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

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

Here "d" is a suitably chosen distance measure. The distance measure "d" used here is taken to be the simple Root Mean Square Error (RMSE) between the original set of gray values and the obtained set of gray values of the concerned range block. The map from $\mathcal{D}_k$ to $\mathcal{R}_i$ is constructed in such a way that the pixel positions of $\mathcal{R}_i$ and $\widehat{\mathcal{R}}_{i|k}$ are the same. The difference between $\mathcal{R}_i$ and $\widehat{\mathcal{R}}_{i|k}$ are found only in the gray level values of the pixels. It may be noted that the theory of Collage theorem [1] has been developed by taking "d" as the Hausdorff metric. Later Jacquin [5] considered "d" to be mean squared error (MSE). The same distance measure has been used in several articles [6,8]. RMSE has been used here for measuring the distortion and it serves the purpose of a distance measure. It has been seen that, for the proposed encoding scheme, attractor - which is close to the original image - may be obtained with the help of RMSE [23].

Now let k be such that

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

The aim here 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 code of image I. Figure 1 illustrates the mapping of domain blocks to the range blocks.

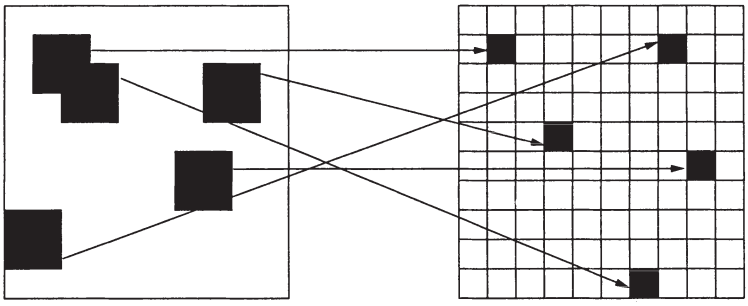

Fig.1. Mapping from Domain block to Range block

To find the best matched domain block as well as the best matched transformation, all possible domain blocks as well as all possible transformations are to be searched with the help of equation (4). The problem of searching for an appropriately matched domain block and transformation for a range block can be solved by enumerative search [5] and by using Genetic Algorithms [10]. We have considered here eight isometric transformations as described by Jacquin [5] to search out the appropriate transformation for a range block. The corresponding parameters of the transformation are selected by fitting a straight line between two sets of gray values, one from range block and the other is from the matched domain block.

2.3 Image Decoding Using PIFS

Let us consider the generated PIFS or fractal code be $\mathcal{F}$ of a given image I. The fractal code $\mathcal{F}$ is obtained through the coding procedure described in Section 2.2. The natural decoding scheme simply consists in iterating the code $\mathcal{F}$ on any initial image, say I_0, until convergence to a stable decoded image is observed. Thus, the decoding process is of the form $\{\mathcal{F}^N(I_0)\}_{N>0}$. At the Nth iteration, the image I_N is used as input to the decoding system, where I_N is the output image obtained from the $(N-1)th$ iteration. The mapping of an image under the fractal code is done sequentially. In particular for each index i, the transformation $f_{i|k}$ is applied to the domain block $\mathcal{D}_k$ of the current input image, and mapped on to the range block $\mathcal{R}_i$ of the current output image which is input image of the next iteration. The sequence of images $\{I_N\}_{N>0}$, is called the reconstruction sequence for the code $\mathcal{F}$ with a starting image I_0. The sequential process is stopped either after a sufficiently large number of iterations or after obtaining a stable image. In our implementation, we have stopped the process of decoding after ten iterations.

3 GAs to Find Fractal Code

We would like to describe first GAs in brief, so that it will be convenient to understand the methodology for finding fractal code using GAs.

GAs are adaptive search processes based on the notion of selection mechanism of natural genetic system [11]. GAs help to find the near global optimal solution without getting stuck at local optima as it deals with multiple points (spread all over the search space) simultaneously. To solve an optimization problem, a GA starts with the structural representation of a parameter set. The parameter set is coded as a string of finite length and the string is called a chromosome. Usually, the chromosomes are strings of 0's and 1's. If the length of string (chromosome) is L then the total number of possible strings is 2^L. 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 [11]. In this chapter, though S has been taken to be an even number, the procedure to be suggested below can be applied to the case of S being an odd number.

Three basic genetic operators, i)Selection, ii)Crossover and iii)Mutation are exploited in GAs. The chromosomes (strings) in an initial pool are judged by their respective fitness values and marked accordingly in selection operation. Next, after forming a new population using selection process, crossover and mutation operators are used one by one to generate a new population of the same size (S). Three operators are then again applied on this new population to give rise to another population. The process of creation of a new population from the existing one is termed as iteration. We have used here the elitist model [13] of GAs which keeps track of the best string obtained so far.

Selection is an artificial version of natural selection mechanism. There are several ways of performing this selection operation. A procedure, which has been followed in this work is called proportional selection strategy [11]. The size or the number of strings in a mating pool is taken to be same as that of the size of the initial population.

There are also several ways of performing the crossover operation [12] among the strings in the mating pool. The single point crossover operation has been followed in this work. The crossover operation is guided by a crossover probability and is represented by P_{cross}.

In the mutation operation every bit of every string is flipped (i.e. 0 by 1 and 1 by 0) with probability P_{mut} which is called the mutation probability. One of the commonly used conventions is to assign a very small value to the mutation probability and keep it fixed for all the iterations, i.e.; the value for P_{mut} is independent of the number of iterations. A different strategy has been adopted here in assigning the value for P_{mut}. We have prefixed the number of iterations of the GA apriori and varied the mutation probability with the number of iterations. This varying mutation probability scheme has already

been applied successfully in connection with an application of GAs to pattern recognition problems [24].

Usually two stopping criteria are used in genetic algorithms. In the first, the process is executed for a fixed number of iterations and the best string, obtained so far, is taken to be the near optimal one. While in the second criterion, the algorithm is terminated if no further improvement in the fitness value for the best string is observed for a fixed number of iterations, and the best string obtained so far is taken to be the near optimal one. In the present work, the number of iterations, say T, is fixed apriori for the termination of GAs. Note that It has been shown that, as the number of iterations in a GA goes to infinity, one can always find the optimal solution [25].

Now let us describe the methodology for finding fractal code using the principle of GAs. The main aspect of fractal based image coding is to find a suitable domain block and a transformation for a range block. Thus the whole problem can be looked upon as a search problem. Instead of a global search mechanism we have introduced GAs to find the near optimal solution.

The number of possible domain blocks to be searched are $(w-2b)\times(w-2b)$ (Section 2.2). The number of transformations to be searched for each domain block is 8 (Section 2.2). Thus the space to be searched consists of M elements. M is called the 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. The value of L depends on the values of w and b. The fitness value of a string is defined 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 form an initial population for GA. The various steps of the GA, as mentioned above 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 range block, and $(M - S\ T)$ provides the reduction in the search space for each rough type range block.

The GA is used here to search for an appropriately matched domain block as well as an appropriate transformation (isometry) for a particular type of range block. The class of range blocks is obtained through a simple classification scheme [10]. Range blocks are grouped into two sets according to the variability of the pixel values in these blocks. If the variability of a block is low *i.e.*, if the variance of the pixel values in the block is below a fixed value, called threshold, we call the block as smooth type range block. Otherwise we call it a rough type range block. After classification, GA based encoding is adopted for rough type range blocks. All the pixel values in a smooth type range block are replaced by the mean of its pixel values.

4 Edge Extraction from Intermediate Representation

Edge extraction using PIFS code is a sequential process. In this sequential process, edges are extracted from the image decoding process (described in Section 2.3). In other words, a decoding process is carried out with some modifications. Now for the better understanding of the process and to know the motivation behind such an algorithm one should know the convergence procedure of PIFS to a stable image through an iterative sequence. The detailed description of the mathematical formulation of the convergence of PIFS is given in [26]. We have furnished here only the relevant portions for convenience.

4.1 Convergence of PIFS

Let I be a given image having size $w \times w$, and the set of gray level values be $\{0, 1, 2, \cdots, l-1\}$. 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 pixels 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 set up PIFS can be viewed as follows. 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. The set $\mathcal{F}$, of forward maps, is called the PIFS codes of I.

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}, where $r_2 = (i_2 - 1)w + j_2$. Thus f_1 is the backward map for x_{r_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.

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

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

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

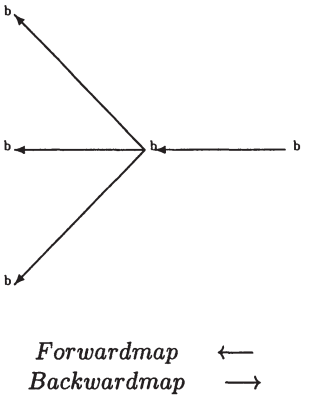

Fig.2. Sketch of Forward map and backward maps

The stopping phenomenon of this sequence is mandatory as there are finite number (w^2) of elements in $\underline{x}$. 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). The detailed description of these four cases and their convergence which is not needed for the present problem, can be found in[26]. The most important thing to be noted is that, for each element, there will be a sequence of the form (5) and each element has fixed points.

4.2 Construction of Edge Image Using PIFS

So far, we were able to establish that, by construction, PIFS code is such that each element of the given image has a sequence of the form (5). Let us consider the number of elements included in a sequence as the length of the sequence. So, from (5) it is quite evident that different elements will have different sequences with different lengths. Now since each of the elements has a fixed point [26], the time taken to converge will be different for different elements. In other words, different elements converge to their respective fixed points at different iterations. The most common question that arises at this juncture is : why the convergence rate is different for different elements. To answer this question, one should recall the construction procedure of the PIFS code. Note that, for the construction of the PIFS code, a range block could be either smooth or rough. PIFS code for a smooth type range block is very simple as every pixel value is replaced by the average of all the pixel

values. So elements (pixels) which are included in the smooth type range block will converge right after first iteration. On the other hand, the edges, which are of main interest here, are included in the rough type range blocks and a GA based fractal encoding scheme is adopted for this type of blocks. In the process of encoding, rough type range blocks will match with domain blocks containing edges. Moreover matching of an edge pixel is restricted to only edge pixels as we are interested in finding the self similarities present in the image. In a rough type range block, in general, there will be both edge pixels and non edge pixels. We want to approximate all the pixels through a contractive continuous map. The closeness of the estimator with a pixel value will depend not only on the map but also on the complexity of the pixel (i.e., the pixel value in relation to its sarrounding pixel values) from where it is being mapped. As one edge pixel is mapped to another edge pixel (since we search for similarities), and, one map is not sufficient to carry the information regarding an edge pixel (since a domain block has edge and non edge pixels), it is expected that an edge pixel is approxiamted by a sequence of maps. Hence, the sequence of the form (5) for an edge pixel will have long length compared to that of a non edge pixel. Thus it is evident that edge pixels will require more iterations to converge compared to non edge pixels. Hence, on the basis of convergence time, we can classify the pixels as edge and non edge.

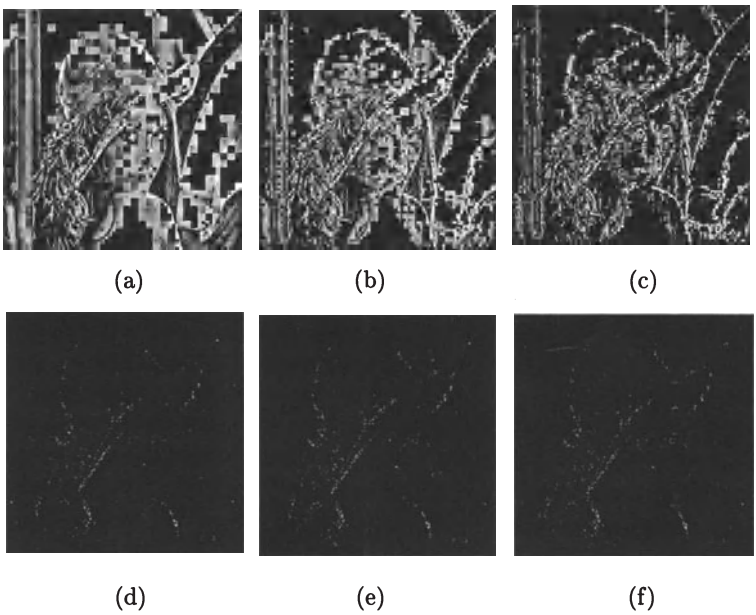

Fig.3. Difference images of "Lena" after 1st (a), 2nd (b), 3rd (c), 8th (d), 9th (e) and 10th (f) iterations with the "Rose" image (Fig. 6.) being the starting image

In support of our claim, we illustrate another phenomenon which occurs during the process of decoding. The decoding process of "Lena" image from its PIFS code has been stopped after ten iterations as no visual difference has been found in successive iterations [10]. But from the difference images, as shown in Fig. 3, it is observed that strong edges are prevelent, though with low intensity, even after ten iterations. These difference images are obtained by taking difference between original image and images obtained during decoding process. This phenomenon experimentaly supports our claim that edge pixels take more time to converge than non edge pixels. Also this illustration indicates the edge information of an image can be extracted from its PIFS code from the reconstruction sequence.

Now the question remaining is: how to extract the edges from PIFS code. To solve this problem, we have introduced a penalty function on the pixels during the process of decoding (convergence) of PIFS code. The reconstruction sequence is modified to include a process of edge extraction. Reconstruction process has now two parts. One is decoding of PIFS code to get an image which is very close to the original one, the other one is the extraction of edges which results in an edge image. The process of edge extraction and the penalty function are discussed below.

We have already mentioned that, edge pixels take long time to converge to their respective fixed points. So, to locate edge pixels we have attached penalties to pixels, depending on their convergence status. The more the number of iteration to converge, the more is the penalty attached to that pixel. Note that, PIFS code is nothing but a set of contractive maps which are continuous. But, edges present in the image are discontinuities due to sudden changes in gray values. During the process of decoding, PIFS code needs to grab those discontinuities by continuous maps. The process of approximating discontinuities by continuous maps is necessarily slow, *i.e.*, it takes more iterations to give rise to a better approximation. Hence, pixels having large penalty values would be edges pixels.

We have designed here a very simple penalty function scheme using the number of iterations, which depends on the status of the convergence. Once the pixel value converges to a fixed point, no further penalty is imposed. On the other hand penalty of each iteration is added with the penalty that already exists for a pixel if the pixel value has not converged yet. The penalty values for the first few iterations need to be small to let the non edge pixels to converge. Similarly, the penalty values for the last few iterations need to be small so that the impact of the strong edge pixels on the other edge pixels would be less. These penalties for the pixels, once the number of iterations for reconstruction is fixed, would indicate the degree to which a pixel may be designated to be an edge pixel. Thus, the edge image is nothing but an image where the gray value of a pixel is nothing but its penalty value. So we will fix two iteration values which will be considered as lower and upper bounds and within these bounds, penalty values will be high. To represent this

phenomenon, an S type penalty function scheme is used. The mathematical formulation of the penalty function is as follows.

Let $p(i,t)$ be the penalty function for a pixel at iteration i. Here t is difference of pixel values between ith and $(i-i)th$ iteration. $p(i,t)$ is defined as follows.

$$\begin{aligned} p(i,t) &= 0 && ;\ i=0\ ,\ \forall t \\ p(i,t) &= p(i-1,t) && ;\ i \geq 1\ ,\ t < \delta \\ &= p(i-1,t) + \sum_{j=1}^{i} j * c(j) && ;\ 1 \leq i \leq N\ ,\ t \geq \delta \end{aligned}$$

where the coefficient $c(j) \in [0,1]$ is computed from a π type function [27] which is as follows

$$\begin{aligned} c(j) &= 1 - 2\left(\frac{|j - \frac{L_1+L_2}{2}|}{L_2 - L_1}\right)^2 \ ;\ 0 \leq j \leq L_1 \quad \text{and} \quad L_2 \leq j \leq N \\ &= 2\left(1 - \frac{|j - \frac{L_1+L_2}{2}|}{L_2 - L_1}\right)^2 \ ;\ L_1 \leq j \leq L_2\ . \end{aligned}$$

Here,

δ = Threshold value of t beyond which penalty is non zero
L_1 = Lower bound for the number of iterations
L_2 = Upper bound for the number of iterations
N = Number of iterations at which sequential process is stopped

These parameters δ, L_1 and L_2 act as control parameters of the proposed scheme. Users can get their desired uotput by adjusting these parameters. For example, if the compression ratio is low one has to fix a large value for L_1, otherwise it may lead to an image with many undesired edges. So, one can play with these parameters and may explore some other interesting results.

Incorporating the penalty function in the reconstruction process, an edge image of the original image can be obtained. Thus, the block diagram shown in Figure 4 provides the overview of the whole process.

Now, it is observed that, in many cases, the output edge image which is obtained from reconstruction sequence with penalty function may have discontinuity in edges. So, we have proposed a very simple edge linking algorithm to get the final edge image.

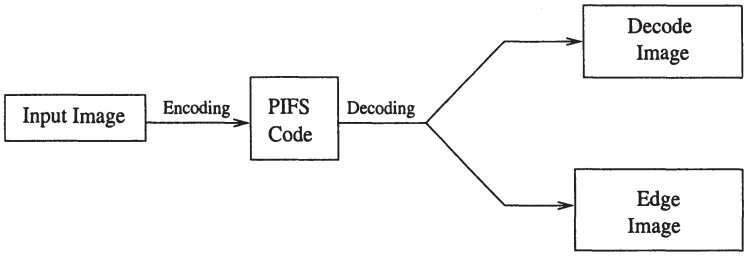

Fig.4. Block diagram of reconstruction process of PIFS code

4.3 Edge Linking

In the above mentioned edge extraction algorithm, the edge image comprises of penalty values associated with each pixel. In this process, some discontinuities may appear at different edge segments. To overcome this we have introduced an edge linking procedure. This edge linking procedure is based on a convolution operation with a set of predetermined kernels (windows) of the same size but having different coefficients. The penalty value of the pixel which is at the center of kernel is modified in course of sliding the kernels over the entire image.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Fig.5. 3 × 3 kernel with coefficients

In particular, we have considered a set of four kernels to find out responses in four directions, namely, horizontal, vertical and two diagonals. The output obtained from the convolution process is called response of a kernel. The maximum response is determined and the center pixel is modified based on this maximum response. For a 3 × 3 window, as shown in Figure 5, the main task involves computing the sum of products of the coefficients (w_i) with the penalty values (p_i) contained in the region encompassed by the window. Thus the response of a kernel for a point at its center is given by

$$w_1p_1 + w_2p_2 + \cdots + w_9p_9 = \sum_{i=1}^{9} w_ip_i \ .$$

Let the response of the kth kernel for its center pixel be denoted by $E_k; k = 1, 2, 3, 4$. The response of a kernel, centered at a boundary pixel, is computed by using the appropriate partial neighborhood. The values of coefficients (w_i) are different for different kernels. For four kernels the values of coefficients in four directions are as follows.

$$\begin{aligned}
\text{Horizontal} &: w_4 = w_5 = w_6 = 1, w_1 = w_2 = w_3 = w_7 = w_8 = w_9 = 0 \\
\text{Vertical} &: w_2 = w_5 = w_8 = 1, w_1 = w_3 = w_4 = w_6 = w_7 = w_9 = 0 \\
\text{First Diagonal} &: w_3 = w_5 = w_7 = 1, w_1 = w_2 = w_4 = w_6 = w_8 = w_9 = 0 \\
\text{Second Diagonal} &: w_1 = w_5 = w_9 = 1, w_2 = w_3 = w_4 = w_6 = w_7 = w_8 = 0
\end{aligned}$$

The choice of the values of the coefficients are similar to that of line detection in four directions [15]. Here p_5 is the penalty value corresponding to the center location of the kernel. Let $E_{max} = MAX_i\{E_i\}$. Compute $A = \frac{E_{max}}{r}$, where, r is the number of nonzero coefficients of the kernel. In the present case $r = 3$ for all kernels. Now the central penalty value (p_5) is modified according to the following rule.

$$\begin{aligned}
p_5 &= p_5 \text{ ; if } p_5 > A \\
&= A \text{ ; if } p_5 \leq A \, .
\end{aligned}$$

The final output edge image is thus obtained from PIFS code by applying a decoding scheme with penalty function (see Section 3) followed by the edge linking algorithm.

The results of the specific implementation of the aforesaid algorithms of image compression and edge extraction are given in the next Section.

5 Implementation and Results

5.1 Image Compression

The GA based method discussed in Section 3 is implemented on 256 × 256, 8 bit/pixel "Lena" image. To make the encoding process faster, the image is subdivided into four 128 × 128 subimages, each of which is encoded separately [5,10]. Also a two level partition [5,10] scheme is adopted for the specific implementation of the GA based Scheme.

First of all, to classify range blocks (Section 3), the variances of pixel values of all 8 × 8 and 4 × 4 range blocks are computed and corresponding thresholds are selected from the respective histograms of the variances. For 8 × 8 range blocks, a valley is found near the value 20 in the histogram and

thus, this value is chosen as threshold for 8×8 range blocks. Similarly, 35 is taken as the threshold value for the 4×4 range blocks.

Considering parent range blocks of size 8×8 and child range blocks of size 4×4 and using two level image partition scheme [5], each subimage is then encoded. GAs 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 cardinalities (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 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 would replace it.

Out of these 2^{17} binary strings, 6 strings ($S = 6$) are selected randomly to construct an initial population. A high probability, say $P_{cross} = 0.85$ is taken for the crossover operation. For mutation operation, P_{mut} (mutation probability) is varied over the iterations and the exact values are 0.30, 0.20, 0.15, 0.10 and 0.06. 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.

For encoding "Lena" image, the results of both GA based method and exhaustive search method are reported in this chapter. Test results and some statistics of the GA based method are given in Table 1.

Table 1. Test results for 256×256, 8 bit/pixel "Lena" image

Type of encoding	Range block size	Domain block size	Number of range blocks				Compression ratio	Bits per pixel	PSNR (in db)
			Parent		Child				
			Smooth	Rough	Smooth	Rough			
Two level	8×8 and 4×4	16×16 and 8×8	278	533	128	951	10.50	0.76	30.22

The diagrams for the original and decoded "Lena" images are shown in figures 7 and 8. Figure 6 shows an arbitrary starting image of "Rose" for the reconstruction of the fractal code of "Lena". The reconstruction sequence is stopped after ten iterations.

The algorithm is also tested for the "Seagull" image. This image is also an 8 bit/pixel image and is of size 256×256. Figures 9 and 10 show the original and decoded (after 10 iterations) "Seagull" images respectively. In this case also, the starting image is the "Rose" image as shown in Figure

6. The compression ratio and PSNR values are found to be 7.40 and 27.27, respectively. Note that the search space reduction ratios for this image are same as that of the "Lena" image since the sizes of parent and child range blocks and the values of S and T are the same.

Fig.6. Rose image

Fig.7. Original Lena

Fig.8. Decoded Lena

Fig.9. Original Seagull

Fig.10. Decoded Seagull

The GA based technique of fractal image compression method is also compared with exhaustive search mechanism. The test results of encoding scheme of both the techniques of "Lena" image are shown in Table 2.

The search space reduction is achieved since near optimal solutions are usually satisfactory and, intuitively, the solutions whose fitness values are far away from the optimal are thrown away in a bulk. This is the reason for GAs to perform well for optimization problems [28].

5.2 Image Edge Extraction

For the specific implementation of the proposed algorithm two types of images are considered. A synthetic image ,"Circle" image as shown in Figure 11, and a real life image, "Lena" image, as shown in Figure 7, are treated as the original input images. The "Circle" image is a 128 × 128, 8 bit/pixel image

Table 2. Results obtained using the GA based technique and Exhaustive search technique for "Lena" image

Type of encoding	Genetic Algorithm			Exhaustive Search		
	Compression Ratio	PSNR in db	Number of domain blocks searched	Compression Ratio	PSNR in db	Number of domain blocks searched
Two level 8×8 & 4×4	10.50	30.22	8102640	11.24	28.32	161869952

and on the other hand "Lena" image is a 256×256, 8 bit/pixel image. Here we have considered a synthetic "Circle" image to judge the performance of the proposed edge extraction algorithm. As there are hardly any measures to judge true edges in a scene, we have tested our algorithm on a synthetic image where the true edges are easily distinguished from non edges by human eyes. The obtained PIFS code by the above mentioned GA based technique is then utilized to extract edges.

Fig.11. Original Circle

Fig.12. Decoded Circle

The obtained PIFS code of "Circle" image is now decoded with the proposed penalty function scheme followed by proposed edge linking procedure with window of size 5×5. Other parameters of edge linking algorithm are set experimentally to get the best results. It has been found experimentally that after 20 iterations most of the PIFS codes get stabled at a fixed point. So, we set $N = 20$. Values of other parameters are $T = 0.1$, $L_1 = 3$ and $L_2 = 17$. Obtained decoded "Circle" image and edge image of "Circle" with edge linking are shown in Figure 12 and Figure 13 respectively. PIFS code of "Lena" is also processed similarly with a window of size 11×11 for edge linking. The resultant edge image of "Lena" is shown in Figure 15.

In practice, the result of edge extraction methods are presented as binary images. Usually, an edge pixel is represented by "1" and non edge pixel is by "0". This process is known as thresholding. To represent the results of the proposed technique in thresholded form, we have set a threshold value "30" and "100" heuristically for "Circle" and "Lena" images respectively. But there are several techniques of thresholding and the users of the proposed technique may choose the methodology of their own choice. The obtained results of thresholded edge image of "Circle" and "Lena" are shown in Figure 14 and Figure 16 respectively.

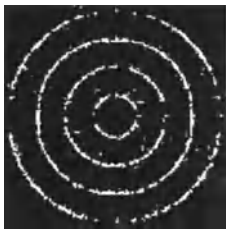

Fig.13. Edge image of Circle

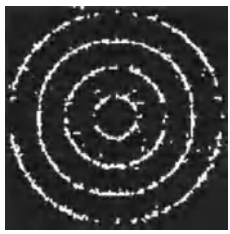

Fig.14. Thresholded edge image of Circle

Fig.15. Edge image of Lena

Fig.16. Thresholded edge image of Lena

6 Discussion and Conclusions

The most important advantage of the proposed technique is that it simultaneously provides techniques of storing images in compressed form and reconstruction of original image or edge image which ever is required. Utilization of the coded version instead of the original image is cost effective in the sense of storage space and time as no separate decoding process is carried out while edge extraction is performed. The performance of proposed edge extraction algorithm is highly dependent on the PIFS or fractal code. It follows that if a fractal code reproduces a good quality reconstructed image then only one

can expect a good quality edge image. So, utmost care should be taken in the encoding process.

The effectiveness of the GA based fractal image compression technique depends upon three factors (i) the number of points in the search space 2^L, (ii) the size of the initial population S and (iii) the number of iterations T. The number of iterations will be different for different images to achieve the near optimal solution using GAs.

The size of the range blocks also plays a vital role in fractal image compression. In particular, finer image detail can be retained by using range blocks of smaller size and more compression can be achieved through range blocks of larger size. While extracting edges it is very important to use the information regarding finer image detail. So, one can think of an optimal range block size for which good quality edge images can be reconstructed from the fractal codes and at the same time considerable amount of compression (in terms of compression ratio) can be achieved. To solve this problem we have used a two level partition scheme [5]. But one can think of quad tree partitioning of the images instead of square partitioning while generating the fractal code [6]. Another scheme [8], in which irregular shapped range blocks are considered, can also be adopted in this connection.

The basic philosophy of the proposed GA based technique can also be adopted to other fractal based coding techniques too. Fisher et al. [6] suggested a quad tree based mechanism for obtaining PIFS code. The GA based method reported in this article and the quad tree based method can be applied simultaneously to give rise to a new method of fractal coding. Quad tree can be applied during the partitioning of the given image and GAs can be applied as the search technique in each step of quad tree partitioning. Thomas et al. [8] suggested an algorithm for fractal based image compression in which the neighborhood information plays an important role in increasing the compression ratio. In that method, the domain block for a "seed" range block is found by an exhaustive search mechanism. The domain blocks for the other range blocks are found by utilizing the connectivity of the range blocks. One can adopt the proposed GA based technique for finding the domain block for the "seed" range block.

To make the proposed GA based fractal image compression technique more efficient in the sense of quality of the decoded image, one can think of a better distortion measure based on the human visual system. PSNR, which is a function of mean squared error (MSE), is used here to measure the quality of the decoded image. There are other measures too to judge the quality of the decoded image [29]. It is easy to modify the proposed method for other measures of judging the quality of the decoded image. Any other function which measures the distortion between the given range block and the obtained range block, can be used as the fitness function instead of MSE in the proposed method. One can achieve a high compression ratio by sacrificing the quality of the decoded image. On the contrary, a high quality decoded

image can be obtained at the cost of compression ratio. Thus, a trade off has to be made to obtain a good quality decoded image with a considerable amount of compression.

In the present article we have suggested a very simple algorithm for edge extraction using fractal code. This algorithm is not the panacea of all the problems of edge detection. The initial results, which are promising, are being reported here. Still there is a large scope of improvement. The design of penalty function, applied in the reconstructed sequence, is very simple. A more generalized penalty function representing an edge pixel in a better way, needs to be found. Such a function may make the edge linking procedure redundant.

The proposed edge extraction technique has a disadvantage, namely, it is noise sensative. Most of the present day edge detector are less noise sensative than the proposed technique. If noise is present in the original image, the fractal code would inherit it and hence the edge image would also be noisy. This problem can only be avoid

References

1. Barnsley, M. F. (1988) *Fractals Everywhere.* Academic Press, New York.
2. Jacquin, A. E. (1989) *Fractal Theory of Iterated Markov Operators With Applications to Digital Image Coding.* PhD thesis, Georgia Institute of Technology.
3. Saupe, D. (1995) "Accelerating fractal compression by multidimensional nearest neighbor search," in *Proceedings of Data Compression Conference*, (Snowbird, UT), 222–231.
4. Jacquin, A. E. (1993) "Fractal image coding : A review," *Proceedings of the IEEE*, **81**, 10, 1451–1465.
5. Jacquin, A. E. (1992) "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Transactions on Image Processing*, **1**, 1, 18–30.
6. Fisher, Y., Jacbos, E. W. and Boss, R. D. (1992) "Fractal image compression using iterated transforms," in *Image and Text Compression* (J. A.Storer, ed.), 35–61, Kluwer Academic Publishers.
7. Boss, R. D. and Jacobs, E. W. (1995) "Archtype classification in an iterated transformation image compression algorithm," in *Fractal Image Compression: Theory and Applications* (Y. Fisher, ed.), 79–90, Springer Verlag, New York, USA.
8. L. Thomas, L. and Deravi, F. (1995) "Region-based fractal image compression using heuristic search," *IEEE Transactions on Image Processing*, **4**, 6, 832–838.
9. Mitra, S. K., Murthy, C. A. and Kundu, M. K. (1995) "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.), 86–91, Narosa Publishing House, New Delhi.
10. Mitra, S. K., Murthy, C. A. and Kundu, M. K. (1998) "Technique for fractal image compression using genetic algorithm," *IEEE Transactions on Image Processing*, **7**, 4, 586–593.

11. Goldberg, D. E. (1989) *Genetic Algorithm in Search, Optimization and Machine Learning.* Addison - Wesley, Reading, Massachussets.
12. Davis, L. (1991) *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.
13. Michalewicz, Z. (1992) *Genetic Algorithms + Data Structure = Evolution Programs.* Springer Verlag, Berlin.
14. Pratt, W. K. (1978) *Digital Image Processing.* John Wiley and Sons Inc., New York.
15. Gonzalez, R. C. and Wood, R. R. (1993) *Digital Image Processing.* Addison Wesley, Massachussets.
16. Canny, J. (1986) "A computational approach to edge detection," *IEEE transaction on Pattern Analysis and Machine Intelligence*, **8**, 6, 679–698.
17. Marr D. and Hilderth, E. C. (1080) "Theory of edge detection," in *Proceedings of Royal Society*, (London Series B 207), 187–217.
18. Haralick, R. M. (1984) "Digital step edge from zero crossing of second directional derivatives," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **6**, 1, 58–86.
19. Kundu, M. K. and Pal, S. K. (1986) "Thresholding for edge detection using human psychovisual response," *Pattern Recognition Letters*, **4**, 433–441.
20. Mitra, S. K., Murthy, C. A. and Kundu, M. K. (1999) "A technique for image magnification using partitioned iterative function system," *Pattern Recognition (Accepted).*
21. McLean, G. E. (1993) "Code book edge detection," *CVGIP:Graphical Models and Image Processing*, **55**, 1, 48–57.
22. Barnsley, M. F. and Hurd, L. P. (1993) *Fractal Image Compression.* AK Press, Massachussets.
23. Mitra, S. K. and Murthy, C. A. (1999) "Mathematical framework to show the existance of attractor of partitioned iterative function systems," *Pattern Recognition (Accepted).*
24. Bandyopadhyay, S., Murthy, C. A. and Pal, S. K. (1995) "Pattern classification with genetic algorithms," *Pattern Recognition Letters*, **16**, 8, 801–808.
25. Bhandari, D., Murthy, C. A. and Pal, S. K. (1996) "Genetic algorithm with elitist model and its convergence," *International Journal of Pattern Recognition and Artificial Intelligence*, **10**, 6, 731–747.
26. Mitra, S. K., Murthy, C. A. and Kundu, M. K. (1998) "A study on partitioned iterative function systems for image compression," *fundamenta Informaticae*, **34**, 4, 413–428.
27. Pal, S. K. and Majumder, D. D. (1986) *Fuzzy Mathematical Approach to Pattern Recognition.* John Wiley (Halsted Press), New York.
28. Murthy, C. A. and Chowdhury, N. (1996) "In search of optimal clusters using genetic algorithms," *Pattern Recognition Letters*, **17**, 8, 825–832.
29. Daly, S. (1992) "The visual difference predictor : an algorithm for the assessment of image fidelity," in *SPIE conference on Human Vision, Visual Processing and Digital Display III*, (San Jose, CA), 2–15.