Asian Journal of Information Technology 7 (7): 285-289, 2008
ISSN: 1682-3915

A Comparative Approach to Fractal image Compression Using Genetic Algorithm and Simulated Annealing Technique

[1]Y. Chakrapani and [2]K. Soundera Rajan
[1]Department of ECE, G. Pulla Reddy Engineering College, Kurnool, A.P., India
[2]Department of ECE, J.N.T.U College of Engineering, Anantapur, A.P., India

Abstract: In this study, the technique of Genetic Algorithm (GA) and Simulated Annealing (SA) is applied for Fractal Image Compression (FIC). With the help of these evolutionary algorithms effort is made to reduce the search complexity of matching between range block and domain block. One of the image compression techniques in the spatial domain is fractal image compression but the main drawback of FIC is that it involves more computational time due to global search. In order to improve the computational time and also the quality of the decoded image, genetic and simulated annealing algorithms are proposed. Experimental results show that the genetic algorithm is a better method than Simulated Annealing Technique for fractal image compression.

Key words: Fractal image compression, genetic algorithm, simulated annealing, gray coding

INTRODUCTION

Compression and decompression technology of digital image has become an important aspect in the storing and transferring of digital image in information society. Most of the methods in use can be classified under the head of lossy compression. This implies that the reconstructed image is always an approximation of the original image. Fractal image coding introduced by Jacquin (1992, 1993), Barnsley (1988) and Barnsley and Jacquin (1998), is the outcome of the study of the iterated function system developed in the last decade. Because of its high compression ratio and simple decompression method, many researchers have done a lot of research on it. But the main drawback of their work can be related to large computational time for image compression. At present, researchers focus mainly on how to select and optimize the classification of the range blocks, balance the speed of compression, increase the compression ratio and improve the quality of image after decompression (Mingshui *et al*., 2004). Especially in the field of reducing the complexity of search, many outstanding algorithms based on classified search have been proposed.

GA is a search and optimisation method developed by mimicking the evolutionary principles and chromosomal processing in natural genetics. Especially GA is efficient to solve nonlinear multiple-extrema problems (Wang and Cao, 2002; Holland, 1975; Goldberg, 1989) and is usually applied to optimize controlled parameters and constrained functions.

Simulated Annealing (SA) is a method for obtaining good solutions to difficult optimization problems which has received much attention over the last few years. The recent interest began with the work of Kirkpatrick and Cerny. They showed how a model for simulating the annealing of solids, as proposed by Metropolis could be used for optimization problems, where the objective function to be minimized corresponds to the energy of the states of the solid. Since then, SA has been applied to many optimization problems occurring in areas such as computer design, image processing and job scheduling. There has also been progress on theoretical results from a mathematical analysis of the method, as well as many computational experiments comparing the performance of SA with other methods for a range of problems.

A few investigations have been carried out in application of GA and SA to fractal image compression. However, these authors have not explained a clear way of applying evolutionary computational techniques like GA and SA to fractal image compression and hence their work does not give a clear idea about the working of GA and SA in FIC. Surprisingly, till date no attempt has been made to show the difference in the performance of GA and SA in fractal image compression.

FRACTAL IMAGE COMPRESSION

The fractal image compression algorithm is based on the fractal theory of self-similar and self-affine transformations.

Corresponding Author: Y. Chakrapani, Department of ECE, G. Pulla Reddy Engineering College, Kurnool, A.P., India

Self-affine and Self-similar transformations: In this study, we present the basic theory involved in Fractal Image Compression. It is basically based on fractal theory of self-affine transformations and self-similar transformations. A self-affine transformation W: $R^n \to R^n$ is a transformation of the form W (x) = T (x) + b, where, T is a linear transformation on R^n and $b \in R^n$ is a vector.

A mapping W: $D \to D$, $D \subseteq R^n$ is called a contraction on D if there is a real number c, $0<c<1$ such that d (W (x), W (y)) $\leq$ cd (x, y) for x, $y \in D$ and for a metric d on R^n. The real number c is called the contractivity of W.

d (W (x), W (y)) = cd (x, y) then W is called a similarity.

A family $\{w_1,....w_m\}$ of contractions is known as Local Iterated function scheme (LIFS). If there is a subset $F \subseteq D$ such that for a LIFS $\{w_1,....w_m\}$:

$$F = \bigcup_{i=1}^{m} w_i(F) \quad (1)$$

Then F is said to be invariant for that LIFS. If F is invariant under a collection of similarities, F is known as a self-similar set.

Let S denote the class of all non-empty compact subsets of D. The δ-parallel body of $A \in S$ is the set of points within distance δ of A, i.e.,

$$A_\delta = \{x \in D: |x - a| \leq \delta, a \in A\} \quad (2)$$

Let us define the distance d (A, B) between 2 sets A, B to be $d(A, B) = \inf \{\delta: A \subset B_\delta \wedge B \subset A_\delta\}$.

The distance function is known as the Hausdorff metric on S. We can also use other distance measures.

Given a LIFS $\{w_1,....w_m\}$, there exists an unique compact invariant set F, such that:

$$F = \bigcup_{i=1}^{m} w_i(F),$$

this F is known as attractor of the system.

If E is compact non-empty subset such that $w_i (E) \subset E$ and

$$W(E) = \bigcup_{i=1}^{m} w_i(F) \quad (3)$$

We define the k-th iteration of W, $W^k (E)$ to be $W^0 (E) = E$, $W^k (E) = W (W^{(k-1)} (E))$.

For $K \geq 1$ then we have that:

$$F = \bigcap_{i=1}^{\infty} W^k(E) \quad (4)$$

The sequence of iteration $W^k(E)$ converges to the attractor of the system for any set E. This means that we can have a family of contractions that approximate complex images and using the family of contractions, the images can be stored and transmitted in a very efficient way. Once we have a LIFS it is easy to obtain the encoded image.

If we want to encode an arbitrary image in this way, we will have to find a family of contractions so that its attractor is an approximation to the given image. Barnsley's collage theorem states how well the attractor of a LIFS can approximate the given image.

Collage theorem: Let $\{w_1,....w_m\}$ be contractions on R^n so that:

$$|w_i (x) - w_i (y)| \leq c|x-y|, \forall x, y \in R^n \wedge \forall i$$

where, $c<1$. Let $E \subset R^n$ be any non-empty compact set. Then,

$$d(E,F) \leq d\left(E, \bigcup_{i=1}^{m} w_i(E)\right) \frac{1}{(1-c)} \quad (5)$$

where, F is the invariant set for the w_i and d is the Hausdorff metric.

As a consequence of this theorem, any subset R^n can be approximated within an arbitrary tolerance by a self-similar set; i.e., given $\delta>1$ there exist contacting similarities $\{w_1,....w_m\}$ with invariant set F satisfying d (E, F)$<\delta$. Therefore, the problem of finding a LIFS $\{w_1,....w_m\}$ whose attractor F is arbitrary close to a given image I is equivalent to minimizing the distance

$$d\left(I, \bigcup_{i=1}^{m} w_i(I)\right)$$

Fractal image coding: The main theory of fractal image coding is based on iterated function system, attractor theorem and Collage theorem. Fractal Image coding makes good use of Image self-similarity in space by ablating image geometric redundant. Fractal coding process is quite complicated but decoding process is very simple, which makes use of potentials in high compression ratio.

Fractal Image coding attempts to find a set of contractive transformations that map (possibly overlapping) domain cells onto a set of range cells that tile the image. The basic algorithm for fractal encoding is as follows:

- The image is partitioned into non overlapping range cells $\{R_i\}$ which may be rectangular or any other shape such as triangles. In this study, rectangular range cells are used.

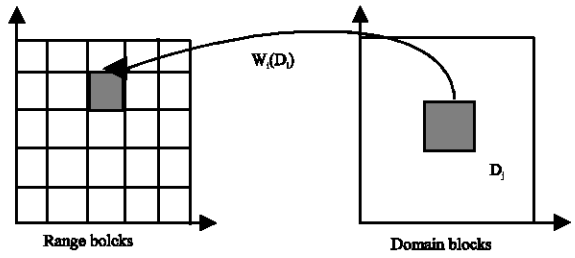

Fig. 1: Domain-Range block transformations

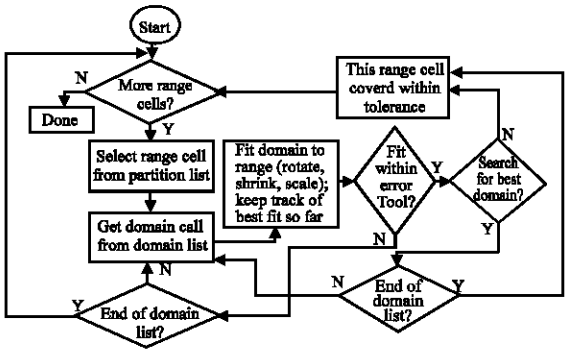

Fig. 2: Flow chart of fractal image encoding

- The image is covered with a sequence of possibly overlapping domain cells. The domain cells occur in variety of sizes and they may be in large number.
- For each range cell the domain cell and corresponding transformation that best covers the range cell is identified. The transformations are generally the affined transformations. For the best match the transformation parameters such as contrast and brightness are adjusted as shown in Fig. 1.
- The code for fractal encoded image is a list consisting of information for each range cell which includes the location of range cell, the domain that maps onto that range cell and parameters that describe the transformation mapping the domain onto the range.

The above fractal image coding algorithm can be represented in the form of flowchart as shown in Fig. 2.

One attractive feature of fractal image compression is that it is resolution independent in the sense that when decompressing, it is not necessary that the dimensions of the decompressed image be the same as that of original image.

GENETIC ALGORITHM

Genetic algorithms are procedures based on the principles of natural selection and natural genetics that have proved to be very efficient in searching for approximations to global optima in large and complex spaces in relatively short time. The basic components of GA are:

- Representation of problem to be solved.
- Genetic operators (selection, crossover, mutation).
- Fitness function.
- Initialization procedure.

GA starts by using the initialization procedure to generate the first population. The members of the population are usually strings of symbols (chromosomes) that represent possible solutions to the problem to be solved as shown in Fig. 3.

Each of the members of the population for the given generation is evaluated and according to its fitness value, it is assigned a probability to be selected for reproduction. Using this probability distribution, the genetic operators select some of the individuals. By applying the operators to them, new individuals are obtained. The mating operator selects 2 members of the population and combines their respective chromosomes to create offspring as shown in Fig. 4. The mutation operator selects a member of the population and changes part of the chromosome as shown in Fig. 5.

The algorithm for the genetic algorithm can be represented as follows:

Step 1: As the genetic algorithm takes pairs of strings, we create a random number of strings depending upon our necessity and also note down their decoded values along with setting a maximum allowable generation number tmax.

Step 2: Using the mapping rule we next find out the corresponding values of above created strings.

Step 3: Using these values the fitness function values are found out.

Step 4: Next the process of reproduction is carried out on the strings to create a mating pool.

Step 5: The process of crossover and mutation is also carried out on the strings with probabilities of 0.8 and 0.05, respectively.

Step 6: After the termination criteria is met with, the value of string with minimum fitness function value is considered as optimum value.

The various parameters of Genetic Algorithm are shown in Table 1.

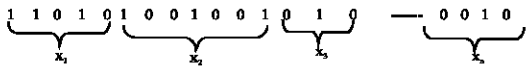

Fig. 3: String representing 'N' variables

Parent A	**00000101**
Parent B	**11101001**
Child A	**00101001**
Child B	**11000101**

Fig. 4: Binary crossover

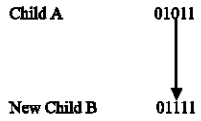

Fig. 5: Binary mutation

Table 1: Genetic Algorithm's parameters

Range block size	4*4	Population size	50
Crossover probability Pc	0.8	Fitness f	1/(1+MSE)
Mutation probability	0.05	Iterations	10

SIMULATED ANNEALING

The simulated annealing method resembles the cooling process of molten metals through annealing. At high temperatures, the atoms in the molten metals can move freely with respect to each another, but as temperature is reduced, the movement of the atoms gets restricted. The atoms start to get ordered and finally form crystals having the minimum possible energy. However, the formation of the crystal mostly depends on the cooling rate. If the temperature is reduced at a very fast rate, the crystalline state may not be achieved at all; instead, the system may end up in a polycrystalline state, which may have a higher energy state than the crystalline state. Therefore, in order to achieve the absolute minimum energy state the temperature needs to be reduced at a slow rate. Simulated annealing is a point-by-point method. The algorithm begins with an initial point and a high temperature T. A second point is created at random in the vicinity of the initial point and the difference in the function values (ΔE) at these 2 points is calculated. If the second point has a smaller function value, the point is accepted; otherwise the point is accepted with a probability exp (-ΔE/T). This completes one iteration of the simulated annealing procedure. In order to simulate the thermal equilibrium at every temperature, a number of points (n) is usually tested at a particular temperature, before reducing the temperature. This algorithm is terminated when a sufficiently small temperature is obtained or a small enough change in function values is found.

Table 2: Simulated annealing parameters

Initial temperature	50,000
No. of iterations at each temperature	25
Termination criterion	2

The algorithm for the simulated annealing can be represented as follows:

Step 1: Choose an initial point x°, a termination criterionϵ, set T a sufficiently high value, number of iterations to be performed at a particular temperature n and set t = 0.

Step 2: Calculate a neighboring point $x^{(t+1)} = N(x^{(t)})$. Usually a random point in the neighborhood is created.

Step 3: If $\Delta E = E(x^{(t+)}) - E(x^{(t)}) < 0$, set t = t + 1. Else create a random number (r) in the range of (0,1). If $r \leq \exp(-\Delta E/T)$ set t = t + 1; Else go to step 2.

Step 4: If $|x^{(t+1)} x^{(t)}| < \epsilon$ and T is small, Terminate, Else if (t mod n) = 0 then lower T according to a cooling schedule. Go to step 2; Else go to Step 2.

The various parameters of Simulated Annealing algorithm are shown in Table 2.

SYSTEM INVESTIGATION

In this study, a gray level image of 256×256 size with 256 gray levels is considered. A Range block of size 4×4 and Domain blocks of size 8×8 are considered. The domain blocks are mapped to the range block by affine transformations and the best domain block is selected. In the case of Genetic Algorithm the individual chromosome is coded as shown in Fig. 6.

The mean squared error (MSE) and PSNR considered in this research are given by:

$$MSE = \frac{1}{N_{Rows} N_{Cols}} \sum_{i=1}^{N_{Rows}} \sum_{j=1}^{N_{Cols}} \left| f_{i,j} - d_{i,j} \right|^2 \quad (6)$$

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) \quad (7)$$

The genetic operators are applied to the string having more fitness value. In the case of Simulated Annealing the temperature value is chosen optimally to find the best suited range block for the domain block.

RESULTS AND DISCUSSION

This research is carried out in MATLAB 7.0 version and the original image is classical 128×128 Lena and Barbara face image coded with 8 bits per pixel. An optimal bit allocation strategy for GA is as follows: 14 bits for the location of matched domain block (horizontal and vertical coordinate), 3 bits for isomorphic types. For each of the range block fractal coding includes 17 bits allocation. During the iteration process of the above proposed methods the gray level values beyond 0 and 255 are replaced by average of its four neighbours to avoid block diverging. Figure 7 and 8 show the reconstructed images using FIC with GA and SA as search algorithms along with the original images of Barbara and Lena after 10 iterations. The Coding scheme of FIC using GA and SA is given in Table 3.

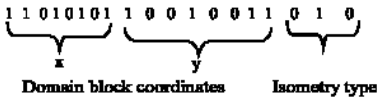

Fig. 6: Binary code for searching domain block

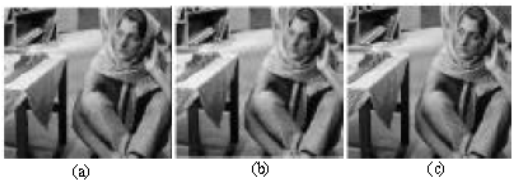

Fig. 7: (a) Original image, (b) Reconstructed image using GA, (c) Reconstructed image using SA

Fig. 8: (a) Original image, (b) Reconstructed image using GA, (c) Reconstructed image using SA

Table 3: Coding scheme comparison with GA and SA

	Image	FIC with genetic algorithm	FIC with simulated annealing
Compression ratio	Barbara	5.0:1	4:1
	Lena	4.5:1	3.5:1
PSNR (db)	Barbara	28.34	23.16
	Lena	26.22	20.32

CONCLUSION

In this study, the concept of GA and SA is applied to FIC. Instead of global searching in FIC the evolutionary computational techniques like GA and SA are implemented which shortens the search space. Experimental results show that the GA scores over SA in the case of fractal image compression. Normally the PSNR ratio for a decoded image should be very high to have a better image. Based on Table 3, it can be seen that the PSNR and Compression ratio are better in the case of decoded image using GA over the one obtained by SA. The performance of GA can be further improved by introducing the concept of elitism which copies the best string in one generation into the second generation.

REFERENCES

Barnsley, M.F. and A.E. Jacquin, 1998. Application of recurrent iterative function systems to images. In: Proc. SPIE, 1001 (3): 122-131.

Barnsley, M.F., 1988. Fractals everywhere. New York: Academic.

Goldberg, D., 1989. Genetic algorithms in search, optimization and machine learning. Addison-Wesley Publishing Company: MA.

Holland, J.H., 1975. Adaptation in nature and artificial systems. University of Michigan Press: Michigan.

Jacquin, A.E., 1992. Image coding based on a fractal theory of iterated contractive image transformations. IEEE. Trans. Image Proc., 1: 18-30.

Jacquin, A.E., 1993. Fractal Image coding: A Review. Proc. IEEE., 81: 1451-1465.

Mingshui Li, Shanhu Ou and Heng Zhang, 2004. The new progress in research approach of fractal image compression. J. Eng. Graph., 4 (3): 143-152.

Wang, X. and L. Cao, 2002. Genetic algorithms-theory, application and software reliability. Xi An Jiao Tong University Press.