Fractal Compression of Single Images and Image Sequences using Genetic Algorithms

Lucia Vences[*]
ITESM - CEM[†]
lvences@servdgi.cem.itesm.mx

Isaac Rudomin[‡]
ITESM - CEM
rudomin@journey.cem.itesm.mx

Abstract

In this paper we present a method to encode a single image by finding an Iterated Function System (IFS) that describes an approximation to the image we want to compress by using Genetic Algorithms (GA). The search was restricted to IFS with a fixed number of maps, and a fixed contractivity factor, like in Barnsley's brute force method.

To improve the convergence for this problem, a modified mating operator was used in the genetic algorithm. By doing this, the time needed to get an IFS is reduced by about 30% compared with Barnsley's brute force method if similar image quality is desired, but more to the point, unlike with other algorithms, using a GA we can vary the time the compression will last as a function of the error we tolerate by varying parameters such as population size and number of generations we allow the algorithm to proceed.

The algorithm was extended to deal with image sequences by using the population that has evolved for an image as the initial population for the next image in the sequence. This did not increase the convergence for those images as we had expected, but further studies are needed before concluding that this will not work.

[*]Graduate Student in the M.Sc. Program in Computer Science.

[†]Instituto Tecnológico de Estudios Superiores de Monterrey, Campus Estado de México.

[‡]Dissertation Advisor.

1 Introduction

The development of a wide range of multimedia applications has recently led to increased research attention in data compression and particullary in image compression.

Among the techniques of image compression, the fractal method based on the theory of Iterated Function Systems (IFS) has recently received a great deal of attention. The basis of this technique, known as *the fractal inverse problem,* is to find an IFS whose attractor is close to a given image.

The main difficulty with this method has been that it takes a long time to compress even a single image. Some methods to reduce the time required to solve inverse problem of a single image have been proposed. One way that has been succesfully used is to apply genetic algorithms (GA). Vrscay [Vrs90] has used GA for moment matching and for the discrete collage method. The results obtained in the first method were not very satisfactory, while the results of the second method were very encouraging. Shonkwiler, Mendivil and Deliu [SMD91] used GA successfully to solve the inverse problem in 1-D and showed that their algorithm can be parallelized making the convergence faster.

In order to use fractal compression on video image sequences, it has been previously proposed that a IFS for the first frame should be constructed, and that for subsequent frames a transformation which takes the previous image to the best possible aproximation to the present image should then be found using other methods [HGB92].

In this paper, we present a technique to compress still images using GA. Due to the temporal and spatial correlations between frames and the application of GA, this method is extended to image sequences. GA are used to obtain, from a randomly generated population, the IFS whose attractor is the best aproximation to a single image or the first frame in the sequence.

When dealing with a image sequence, once we have obtained a code for the first frame, for the subsequent frames we can use the last population generated for the previous frame as the initial population for the present frame; in this way, we hope to have an even faster convergence for the compression of the following frames.

In the next section, we present the theoretical basis for fractal image compression, the collage theorem wich is the key for this technique, some of the algorithms that have been used and general background about GA.

The third section is dedicated to the details of our algorithm: the kind of genes used, the genetic operators applied and some others improvements to the simple genetic algorithm.

In the results section, we examine the performance of our algorithm for both a single image and image sequences.

In the conclusion section, we give some directions for further work.

2 Theoretical Foundations

In this section we expose the theoretical basis for fractal image compression and that the basis of genetic algorithms.

2.1 Self-affine and self-similar transformations

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

A self-affine transformation $W : \Re^n \rightarrow \Re^n$ is a transformation of the form $W(x) = T(x) + b$, where T is a linear transformation on $\Re^n$ and $b \in \Re^n$ is a vector. A mapping $W : D \rightarrow D, D \subseteq \Re^n$ is called a contraction on D if there is a real $c, 0 < c < 1$, such that

$$d(W(x), W(y)) \leq cd(x, y)$$

for $x, y \in D$ and for a metric d on $\Re^n$. If

$$d(W(x), W(y)) = cd(x, y),$$

then W is called a *similarity*. The real c is called the contractivity of W.

A family $\{\omega_1, \ldots, \omega_m\}$ of contractions is known as *iterated function scheme or system* (IFS). If there is a subset $F \subseteq D$ such that for an IFS $\{\omega_1, \ldots, \omega_m\}$,

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

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

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

$$A_\delta = \{x \in D : |x - a| \leq \delta, a \in A\}.$$

Let us define the distance $d(A, B)$ between two sets A, B to be

$$d(A, B) = inf\{\delta : A \subset B_\delta \wedge B \subset A_\delta\}.$$

This distance function is known as the *Hausdorff metric* on $\mathcal{S}$[1].

Given an IFS $\{\omega_1, \ldots, \omega_m\}$, there exists an unique compact invariant set F, such that

$$F = \bigcup_{i=1}^{m} \omega_i(F).$$

This F is known as the attractor of the system.

If E is a compact non empty subset such that $\omega_i(E) \subset E$ and

$$W(E) = \bigcup_{i=1}^{m} \omega_i(E),$$

[1]Other distance functions can be used.

we define the k-th iteration of $W, W^k(E)$, like

$$W^0(E) = E, W^k(E) = W(W^{k-1}(E)),$$

for $k \geq 1$, then we have that

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

2.2 The collage theorem

At this point, we have that the sequence of iterations $W^k(E)$ converges at 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 this family of contractions, they can be stored and transmitted in a very efficient way. Once we have an IFS, it is straightforward to obtain the image encoded by it.

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 an IFS can approximate the given image.

Collage Theorem. Let $\{\omega_1, \ldots, \omega_m\}$ be contractions on $\Re^n$ so that

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

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

$$d(E, F) \leq d(E, \bigcup_{i=1}^{m} \omega_i(E)) \frac{1}{(1-c)}$$

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

As a consequence of this theorem, any subset of $\Re^n$ can be approximated within an arbitrarily tolerance (or error) by a self-similar set; i.e., given $\delta > 0$, there exist contracting similarities $\{\omega_1, \ldots, \omega_m\}$ with invariant set F satisfying $d(E, F) < \delta$.

Therefore, the problem of finding an IFS

$$\{\omega_1, \ldots, \omega_m\}$$

whose attractor F is arbitrary close to a given image I is equivalent to minimizing the distance

$$d(I, \bigcup_{i=1}^{m} \omega_i(I)).$$

2.3 Fractal image compression algorithms

These ideas were originally developed by M. F. Barnsley [BaSl88, BaHu92, HGB92] to encode real world images.

The algorithm he proposes works by dividing the image into a grid of non-overlapping domain blocks $D = \{D_i\}$ of $B \times B$ pixels and into arbitrary range blocks $R = \{R_i\}$ of $2B \times 2B$ pixels and finding the IFS from a range block to a domain block. For each R_i, a search for the best D_i over all the set D must be performed.

One nice feature of fractal image compression is that it is resolution independent in the sense that when decompressing, the dimensions of the decompressed image are not required to be the same as those of the original image because what is stored encodes the transformations required to transform any image to an approximation of the original image.

The size of the transformed blocks is not part of the encoding, and so can be changed. If the image is decompressed to a larger size than that of the original image detail is generated in a way that is acceptable to the eye.

Barnsley's algorithm works well when it is parallelized or is encoded in hardware; but, in general, its temporal complexity is very high, i.e. it takes a long time to compress an image.

Jaquin [Jaq92] introduced a categorization of the blocks in *edge* , *midrange* and *shade* blocks, so that the search for a range block is done only on the domain blocks of the same category. This gives the algorithm a much better performance by reducing the search universe for each block. This has prompted many other approaches to somehow reduce the search space to blocks that can be thought of as similar in some way:

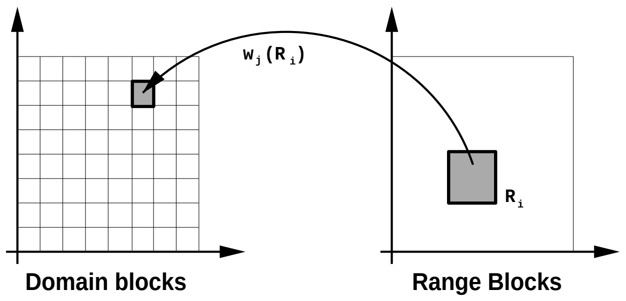

Figure 1: Barnsley's image compression algorithm

- Fisher et. el. [FiJB92] uses a variable number of classes taking into acount variance as well as intensity to classsify a block.
- Saupe [Saup94b] uses a multidimensional nearest neighbour (*kd*-tree based search that runs in logarithmic time instead of linear time.

Other methods have been published that increase the performance of the fractal compression algorithms. A recent review is given by Saupe [Saup94].

Other people have investigated the use of GAs for the fractal image compression problem (or similar problems), although not necesarilly based on Barmsley's method.

- Vrscay [Vrs90] analized the problem for the moment method. It is known that the moments are invariants of a measure; that is, two IFS have the same measure if, and only if, they have the same moments. Vrscay's idea was to minimize the distance between the moments of an IFS and the image. To minimize this distance, simulated annealing or gradient descent methods have been used. Vrscay applied GA to the minimization of moment distance functions with what he calls unsatisfactory results. On the other hand, he also applied GA to the discrete collage method with very promising results.
- Around the same time, R. Shonkwiler, et al [SMD91], reported good results when applying GA to the solution of the inverse problem on 1-D with parallel computation. They reported that the algorithm was much faster when it was parallelized.

2.4 Genetic algorithms

Genetic Algorithms (GA) are search procedures based on the principles of natural selection and natural genetics [Gol89], that have proved to be very efficient searching for approximations to global optima in large and complex spaces in relatively short time.

A GA has as a basic components some genetic operators, an appropriate representation of the problem for these operators, a fitness function and an initialization procedure.

With these basic components, a GA works as follows. It starts using the initialization procedure to get the first population. Each of the elements of the generation is evaluated, and, according with the fitness each element gets, it is assigned a probability to be selected for reproduction.

Using this probability distribution, the genetic operators select some of the individuals of the generation to be applied on them and so obtain new individuals.

The populations' elements with the worst fitness are replaced by the new individuals. The algorithm continues working until some termination criteria is satified.

In fractal image compression, GA's have been used to find solutions to the minimization problems related to the fractal inverse problem [Vrs90, SMD91].

3 Genetic Algorithm methods for fractal image compression

In this section we describe the GA we used to solve the fractal inverse problem. The basic structures are explained, as well as how the genetic operators were defined and how they work.

3.1 Genes

The encoding of the maps are to be used as the genes of the GA, i.e. the IFS. The maps we use are like the ones of Jaquin [Jaq92]. We store four parameters for each block: the x,y coordinates for the corresponding domain block, the isometry to be appied and a luminance shift.

We applied a fixed contraction factor to all the domain blocks; the isometric transformations were the eight canonical isometries of a square block. They are summarized in table 1.

In the algorithm that we have developed, algorithm, we use (as Vrscay [Vrs90] and Shonkwiler et al [SMD91] did) a population of IFS maps of fixed length. Therefore the compression ratio is constant.

In fact since we use the same encoding scheme as the one we use in our implementation of Barnsley's algorithm, the compression ratio of our method will be exactly the one that we get for Barnsley's method.

The image to be encoded was divided into non-overlapping $B \times B$ domain blocks $D = \{D_i\}$: the length of the genes is therefore determined by the size of the image.

#	Isometry	Formula
0	Identity	$(i_0\mu)_{i,j} = \mu_{i,j}$
1	Orthogonal reflection (vertical axis)	$(i_1\mu)_{i,j} = \mu_{i,B-1-j}$
2	Orthogonal reflection (horizontal axis)	$(i_2\mu)_{i,j} = \mu_{B-1-i,j}$
3	Orthogonal reflection (first diagonal) $(i = j)$	$(i_3\mu)_{i,j} = \mu_{j,i}$
4	Orthogonal reflection (second diagonal) $(i + j = B - 1)$	$(i_4\mu)_{i,j} = \mu_{B-1-j,B-1-j}$
5	Rotation (center of block) $+90°$	$(i_5\mu)_{i,j} = \mu_{j,B-1-i}$
6	Rotation (center of block) $+180°$	$(i_6\mu)_{i,j} = \mu_{B-1-i,B-1-j}$
7	Rotation (center of block) $-90°$	$(i_7\mu)_{i,j} = \mu_{B-1-i,j}$

Table 1: Isometric transformations

3.2 Genetic operators

We applied two genetic operators: a mating operator and a mutation operator. The mate is performed by selecting two elements in the present population as the parents of new elements. This election of the parents is made according to a probability distribution that is inversely related to the $\mathcal{L}$–2 distance between the attractor of the IFS and the image[2].

Since all the elements have the same length and the same representation, the crossover point can be anywhere along the gene.

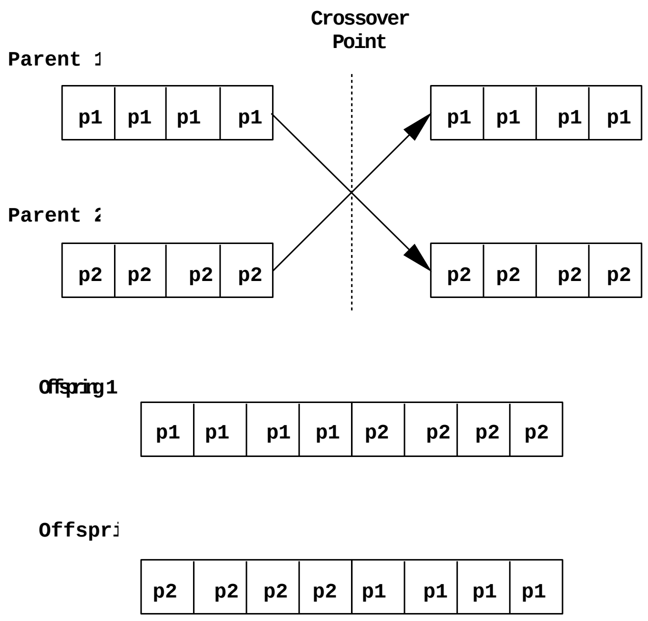

Figure 2: Mating operator for the simple GA

When using a simple genetic algorithm, the two parents combine the two parts in wich they are divided, before the crossover point and after it, to create two new elements: one with the first part taken from the

[2]The $\mathcal{L}$–2 distance is defined as

$$\delta_{\mathcal{L}-2}(\mu, \hat{\mu}) = \sum_{0 \leq i,j < B} (\mu_{ij}, \hat{\mu}_{ij}).$$

first parent and with the second part taken from the second parent, and the other in exactly the opposite way. See figure 2.

In order to have a faster convergence, we changed this mating operator. Each of the maps forming a gene in the population corresponds to a well determined block and its performace is independent of the fitness of any other map in the gene. We took advantage of this fact in the mating operation, so that when a couple of parents are selected, we obtain an offspring just by taking the best suited map for each block. See figure 3.

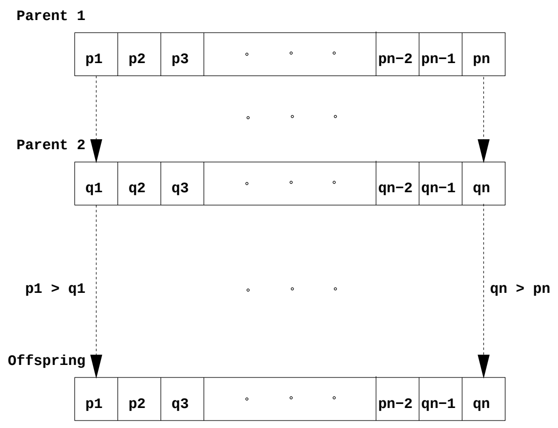

Figure 3: Modified mating operator (here $p_i > q_i$ means that the image for block i is better approximated by the IFS encoded by p_i than by that encoded by q_i).

Notice that although mate selection is still somewhat random, the modified mating algorithm itself is deterministic. In our experiments with a single image we observed that the convergence was significantly accelerated with this modified mating operator. The determinism, however might cause premature convergence thus reducing the variety in the population. This might explain some of the problems with the extension of the algorithm to image sequences[3].

The mutation operator can be randomly applied to any parameter of a gene while creating a new element of the population.

We decided to modify the mutation operator as well, by obtaining a random change around the previous values instead of a random value. This accelerated slightly the convergence in our experiments.

3.3 Fractal image sequence compression

With the algorithm described in the previous sections we obtain a procedure to encode still images.

Compression on image sequences can be achieved by taking advantage of the the spatial and temporal correlations that exist between consecutive frames.

To compress image sequences, it has been previously proposed [HGB92] that an IFS for the first frame should be found. After that, a transformation mapping the previous frame to the present frame is found (not an IFS).

With the technique we are presenting, once we have obtained the "best" IFS for a frame, in the rest of the population we have a set of IFSs that can be used as the *initial population* with a new target for the GA; this new target is the following frame in the sequence. We expected the convergence to this new goal to be faster than if we were dealing with a random initial population, but in fact this did not happen.

4 Results

All tests here described were performed on an IBM-RS600 model 375. The test image was a 296 by 240 grey-scale image. For both the Genetic Algorithm and Exhaustive Search, domain blocks were 8 by 8 pixels[4]

[3]Other mating schemes might be appropriate that avoid these problems and should be studied, among them considering the coding of each domain block separately (as in speciation) using the standard GA for each of them.

[4]This was chosen arbitrarily. It seemed to us that it provided decent quality and significant compression. Better images would result from taking 4 by 4 blocks, but the compression ratio would

and , so the compression attained was the same for all methods.

Differences in resulting images were therefore in quality and/or time required to achieve the coding. Error was measured as distance between the resulting image and the original image.

4.1 Results for a Single Image

Several tests were made using the simple mating algorithm, for a fixed number of generations using both the $\mathcal{L}^1$ and $\mathcal{L}^2$ metrics to see which would be a better guide for the GA. The $\mathcal{L}^2$ metric appeared to give smaller errors, so it was chosen for the following tests.

To select the population size populations of size 512, 256, 128 and 64 were tried. As we can see in figure 4, it turned out that a size of 64 was slightly worse than the others, among which no real differences were found. Since the smaller populations give a faster algorithm, the population size chosen was 128.

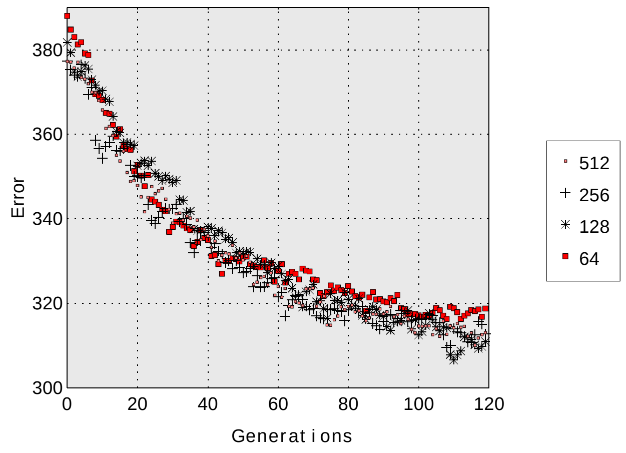

Figure 4: Simple GA with different population sizes: 512, 256, 128, 64.

In the same figure you can see that convergence of the algorithm is slow, even compared with Barnsley's exhaustive search method. Since Barnsley's method will find the best approximation possible with the partition chosen, and since we are using the same partition for the GA, we can't get a better approximation or compression ratio, all we can do is to try to be faster than exhaustive search.

The following tests were performed using the modified mating algorithm described above. As it can be seen in figure 5 convergence was significantly accelerated. Using similar time periods as that necessary for Barnsley's exhaustive search, we get similar quality. The important fact, however is that to get a comparable image quality we can use less time if so desired, particularly since this figure shows that the error drops rapidly at the beginning (in less than 20 generations) to acceptable levels, and keeps on converging later, although much more slowly.

This means that we don't really have to use more than 20 generations to get decent quality. For an ilustration see figures 6, and 7.

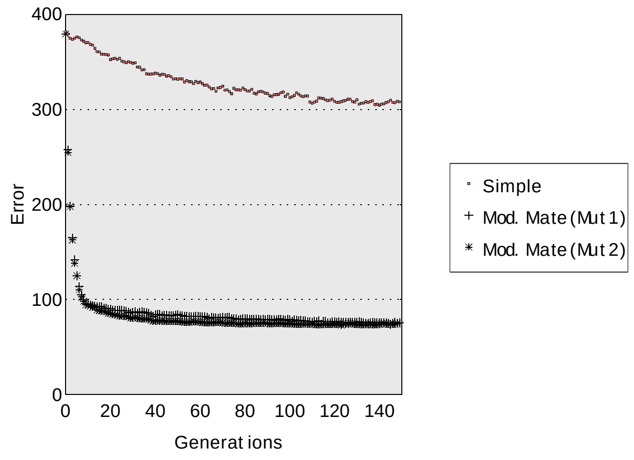

Figure 5: Comparison between the different algorithms.

In table 1 the time and error for the GA methods used are compared.

be smaller.

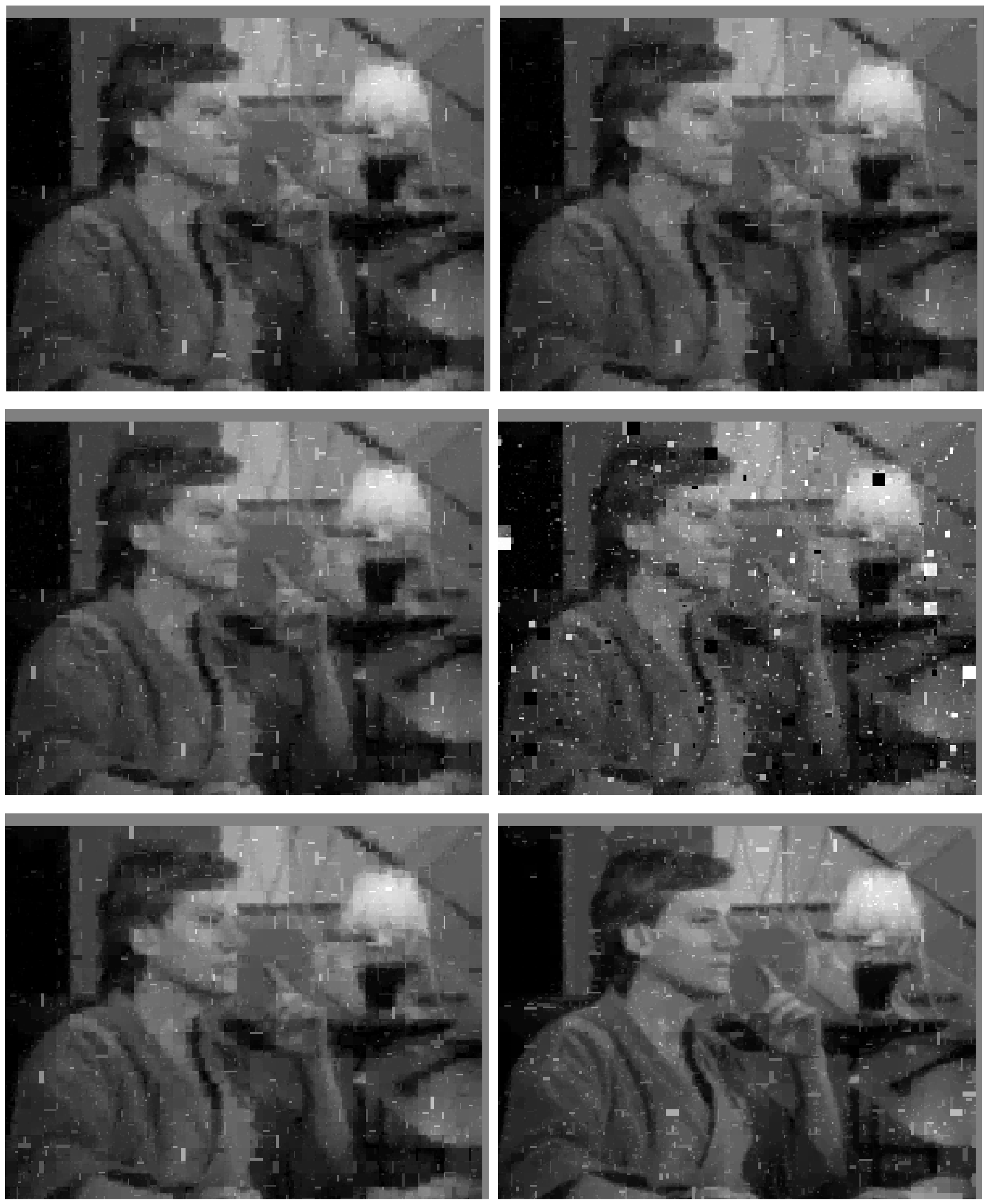

Figure 6: Images obtained by the GA in generations 0, 6, 12, 18, 24, 30.

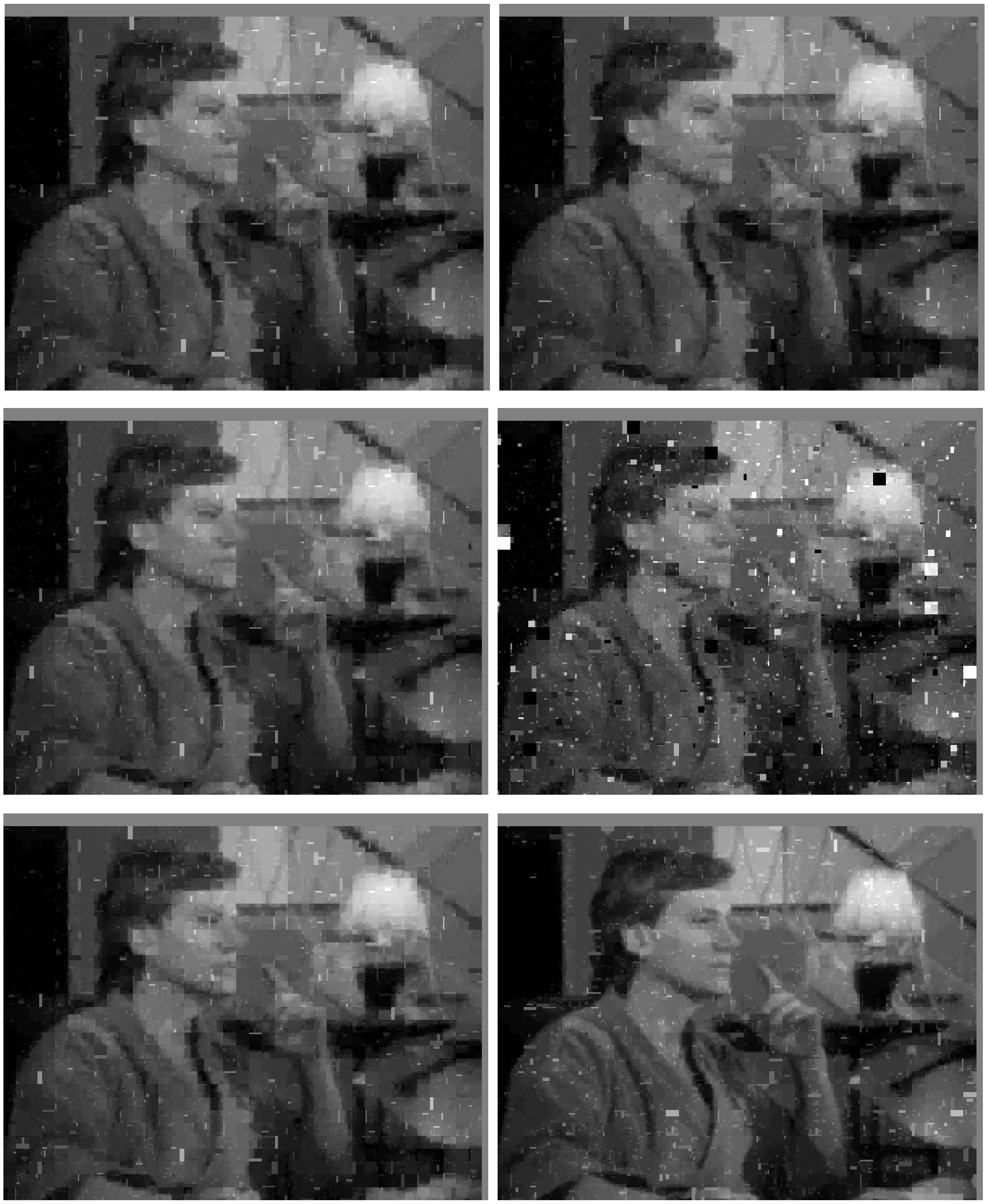

Figure 7: Images obtained by the GA in generations 36, 42, 48, 54, 60 and exhaustive search.

Time (min)	*Barnsley*	*simple GA*	*Modified Mating*	*Modified Mutation*
33	- -	358.03	82.33	80.37
82.5	- -	342.39	76.98	69.64
115	- -	321.48	71.94	66.01
247.5	- -	315.70	68.78	65.48
330	40.29	312.74	66.85	65.45

Table 1: Time and Error in the different methods.

Other tests used a modified mutation operator, but the results did not change much.

In figure 8, we show the images obtained by the different algorithms used.

4.2 Results for Image Sequences

Several tests were performed in image sequences using the modified mating operator and both mutation operators. The algorithm compresses the first image of the sequence as previously done for a single image, by starting from a random population and looking for a IFS for the image by using a GA.

When one is found, the population is stored and used as the initial population for the second image, and so on for subsequent images. The initial error obtained when using the stored final population for the previous image as the initial population is indeed smaller than that found for a random initial population, but the convergence is slower and the algorithm using a random initial population overtakes the one using that stored initial population.

This can be seen in figures 9 and 10. The second mutation operator is better than the other, but the convergence is still slower than that found in either case by using the new random population.

This behaviour suggests that the stored population is overspecialized. Several tricks were tried without success to overcome this problem:

- We tried using a mixed initial population: partly that which was stored, partly a random population. At best it performed like a random population.
- We tried increaing the probability of mutation. This caused more variability as desired, but decreased convergence significantly.

The problem might have to do with the deterministic nature of the modified mating operator and so must be further studied. In any event even the algorithm as used for a single image can be used to compress each image in the sequence separately, and the performance gains found for a single image will also apply to the sequence.

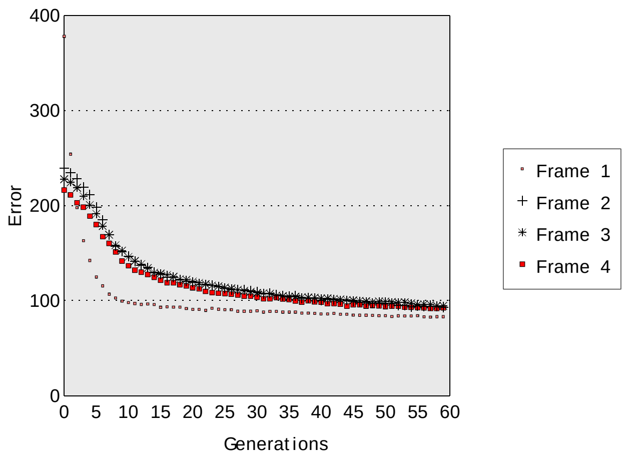

Figure 11: Errors for Image sequence of four consecutive video frames. The second mutation operator and stored initial populations were used.

5 Conclusions

We have presented a GA for fractal image compression both with a simple mating operator and with a modified mating operator. The changes made to the simple GA have improved the performance of the technique leading to a faster convergence for the fractal inverse

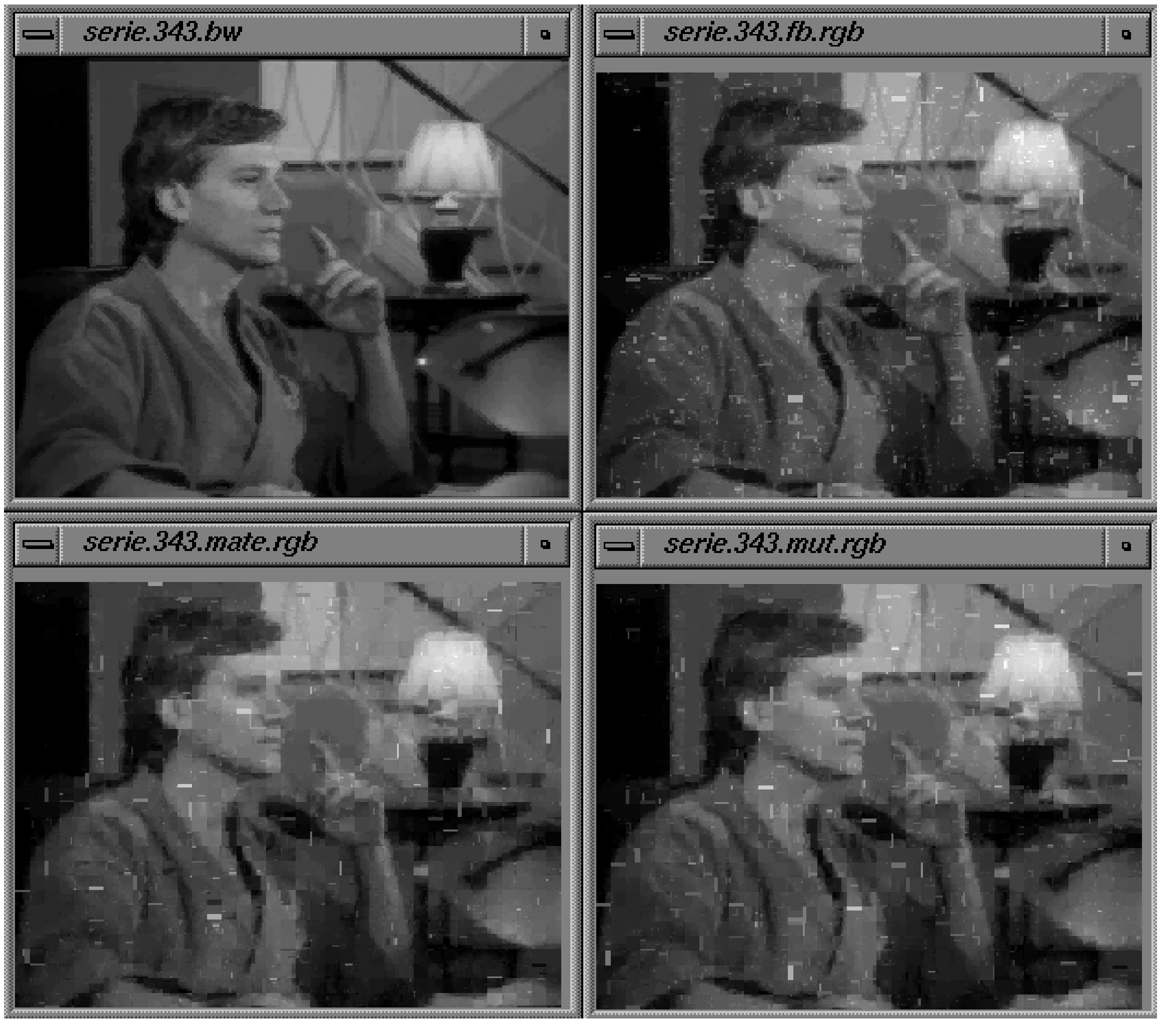

Figure 8: Original Image (above left) compared with the images obtained by using Barnsley's algorithm (above right) and GA's using modified mating operator (below left) and modified mating and mutation operators (below right).

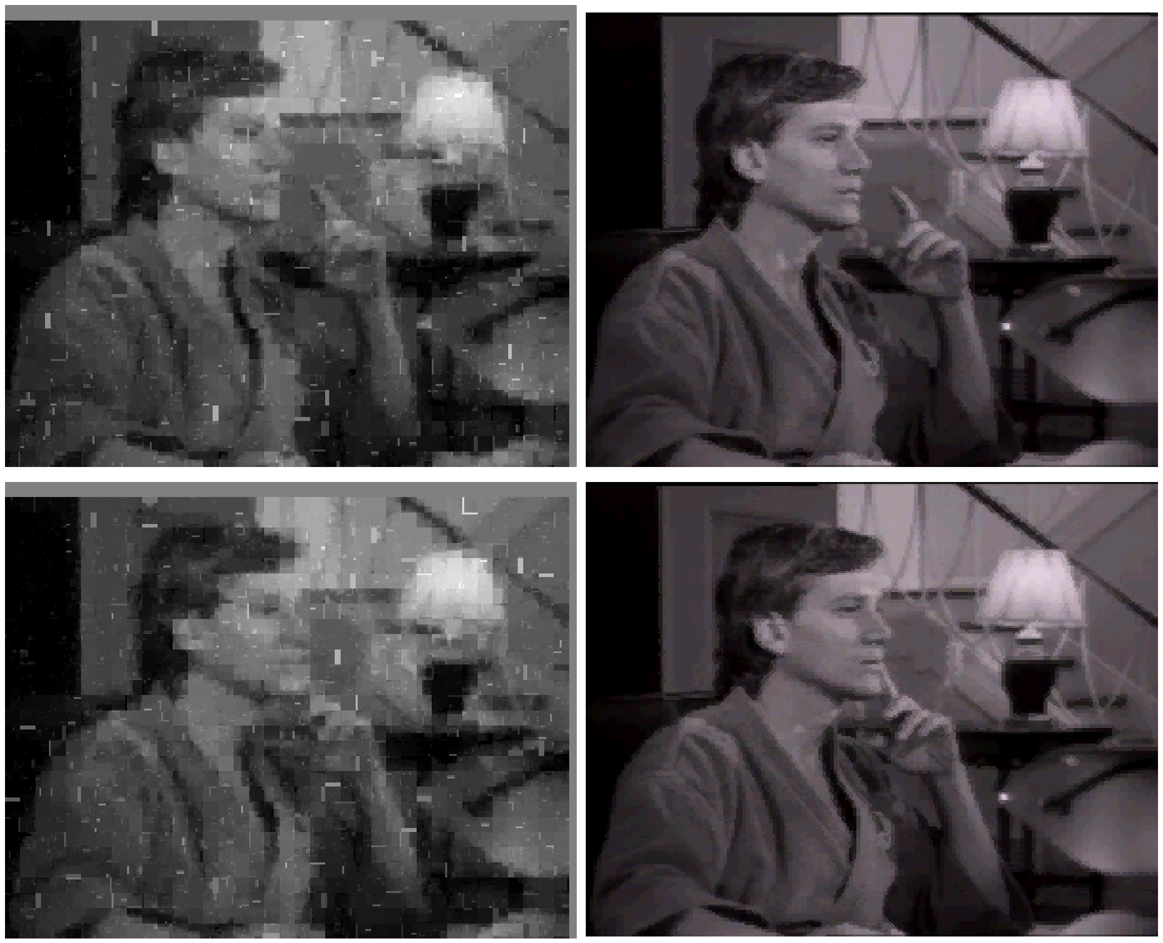

Figure 9: Image sequence: first two of four consecutive video frames, in compressed and original form. The second mutation operator and stored initial populations were used.

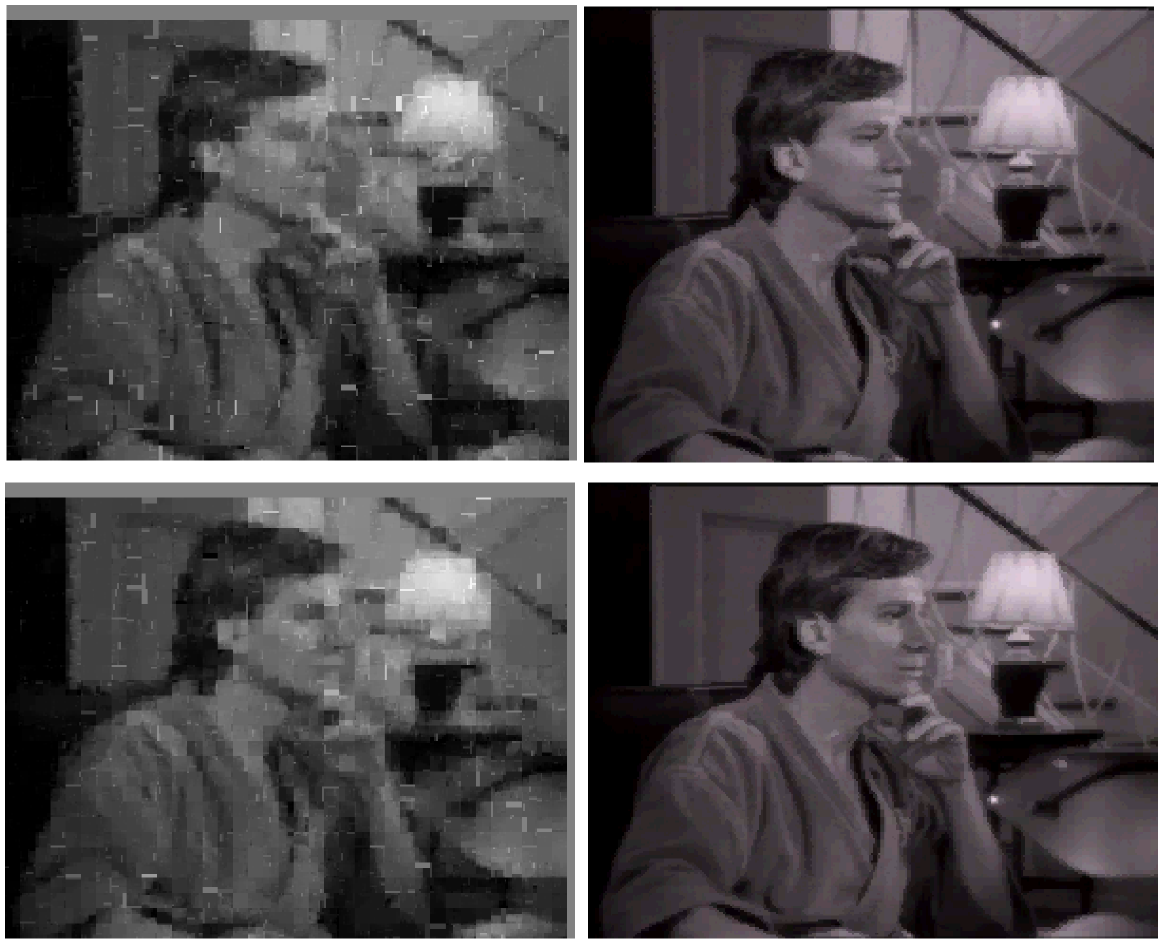

Figure 10: Image sequence: last two of four consecutive video frames, in compressed and original form. The second mutation operator and stored initial populations were used.

problem. For a single image, a 30% speedup compared with the exhaustive search algorithm was found.

For image sequences the expected faster convergence was not found, but it seems the population was overspecialized. Further work must be performed to avoid this overspecialization. However, even using the algorithm that restarts with a random population for each subsequent image, we will benefit from the performance gains found for the algorithm for a single image.

Other results in fractal image compression that give better results than exhaustive search could and should be incorporated within this GA strategy: in particular, using different partitions (like quadtrees) and classification schemes (like edge/midrange/shade blocks) within the GA framework could be useful. Incorporating some of the more recent ideas (hierarchical codification, more efficient data structures for the search) might also prove useful.

References

[BaSl88] Barnsley, M.F., Alan D. Sloan, *A Better Way to Compress Images*, BYTE, January, 1988.

[BaHu92] Barnsley, Michael, Lyman P. Hurd, *Fractal Image Compression*, AK Peters, Ltd (1992).

[Fis92] Fisher,Yuval, *Fractal Image Compression*, SIGGRAPH 92, Course Notes.

[FiJB92] Fisher, Y., Jacobs, E. W., Boss, R. D., *Fractal image compression using iterated transforms*, in: J. A. Storer (ed.), *Image and Text Compression*, Kluwer Academic Publishers, Boston 1992.

[Gol89] Goldberg E.David, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Massachusetts (1989).

[HGB92] Hurd, L.P., M.A. Gustavus, M.F. Barnsley, *Fractal Video Compression*, Digest of Papers, COMPCON Spring 1992, 37th IEEE Computer Society International Conference, p.41-2, 1992.

[Jaq92] Jacquin, Arnaud E, *Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations*, IEEE Transactions on Signal Processing, March 1992.

[MaMo92] Mazel, David, Monson Hayes, *Using Itereted Function Systems to Model Discrete Sequences*, IEEE- Transactions on Signal Processing 40, July 92.

[MoDu92] Monro, D.M., F. Dudbridge, *Fractal Block Coding of Images*, IEEE Electronic Letters, 21st May 1992, Vol.28, No.11.

[Mon93] Monro, D.M. *Class of Fractal Transforms*, IEEE Electronic Letters, 18th February, 1993, Vol.29, No.4

[Saup94] Saupe, D *A Guided Tour of the the Fractal Image Compression Literature*, Technical Report, Institut für Informatik, Universität Freiburg, 1994, available from ftp://fidji.informatik.uni-freiburg.de/papers/fractal/Guide.ps.Z

[Saup94b] Saupe, D *Breaking the Time Complexity of Fractal Image Compression*, Technical Report, Institut für Informatik, Universität Freiburg, 1994, available from ftp://fidji.informatik.uni-freiburg.de/papers/fractal/Saup94a.ps.Z

[SMD91] Shonkwiler, R, F. Mendivil, A. Deliu, *Genetic Algorithms for the 1-D Fractal Inverse Problem*, Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego (1991).

[Vrs90] Vrscay, Edward R. *Moment and Collage Methods of the Inverse Problem of Factal Construction with Iterated Function Systems*, Proceedings of 1st IFIP Conference on Fractals, FRACTAL 90, Lisbon (June 6-8, 1990).