



P-2838



# **FAST FRACTAL IMAGE COMPRESSION BASED ON ENTROPY**

**A PROJECT REPORT**

**Submitted by**

**GUNAPRIYA.K**

**71205104010**

**HEMASREE.S**

**71205104014**

*in partial fulfilment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**

**ANNA UNIVERSITY, CHENNAI 600 025**

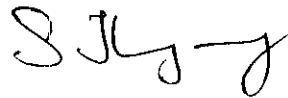
**APRIL 2009**



ANNA UNIVERSITY, CHENNAI 600 025

**BONAFIDE CERTIFICATE**

Certified that this work report "FAST FRACTAL IMAGE COMPRESSION BASED ON ENTROPY" is the bonafide work of GUNAPRIYA.K, HEMASREE.S who carried out the project work under my supervision.



**SIGNATURE**

Dr.S.Thangasamy,

**DEAN,**

Dept of CSE,

Kumaraguru College of Technology,

Coimbatore-641006.



**SIGNATURE**

Mr.M.Muthukumar,

**SUPERVISOR,**

Lecturer,

Dept of CSE ,

Kumaraguru college of Technology,

Coimbatore-641006.

The candidates with University Register Nos. **71205104010**, **71205104014** were examined by us in the project viva-voce examination held on. 27.04.09



**INTERNAL EXAMINER**



**EXTERNAL EXAMINER**

**DECLARATION**

We,

GUNAPRIYA.K            71205104010

HEMASREE.S            71205104014

hereby declare that the project entitled "**FAST FRACTAL IMAGE COMPRESSION BASED ON ENTROPY** " , submitted in partial fulfillment to Anna University is a record of original work done by us under the supervision and guidance of the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

Date: 27/4/09

  
[GUNAPRIYA.K]

  
[HEMASREE.S]

## ACKNOWLEDGEMENT

We would like to express our gratitude to **Prof.R. Annamalai M.E.**, Vice Principal, Kumaraguru College of Technology, Coimbatore for helping us to complete this work successfully.

We are overwhelmingly grateful to **Dr.S.Thangasamy Ph.D.**,Dean, Department of Computer Science and Engineering, Kumaraguru College of Technology, for his valuable guidance and useful suggestions.

We have immense pleasure in expressing our heartfelt thanks to our guide **Mr.M.Muthukumar**, Lecturer, Computer Science Department, Kumaraguru College of Technology, for his constant advice and support during the working of the project. We are grateful to him for his guidance.

We would like to thank our project co-ordinator **Mrs.P.Devaki** , Assistant Professor, Computer Science Department, Kumaraguru College of Technology, for her support during the course of the work.

We would like to express our sincere thanks to all the **teaching and non-teaching staff** of the Department of Computer Science and Engineering for their support both directly and indirectly.

We also thank our **parents and friends** for the support for the completion of our work successfully.

## CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vii
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
2	OVERVIEW	4
	2.1 SPECIAL FEATURES	5
	2.2 CONSTRAINTS	5
3	LITERATURE REVIEW	6
4	PROPOSED SYSTEM	9
	4.1 ERROR METRICS	11
	4.2 ROTATION, MIRRORING AND SCALING	12
	4.3 SOFTWARE REQUIREMENTS	14
	4.3.1 HARDWARE CONFIGURATION	14
	4.3.2 SOFTWARE CONFIGURATION	14
5	IMPLEMENTATION	15
	5.1 PARTITIONING OF THE IMAGE	16

	5.2 ITERATIVE FUNCTION SYSTEM	17
	5.3 FRACTAL ENCODING	18
	5.4 ENTROPY EVALUATION	20
	5.5 FRACTAL DECODING	21
	5.6 ALGORITHM FOR FINDING THE BEST MATCH	23
6	CONCLUSION	24
7	FUTURE ENHANCEMENTS	26
8	REFERENCES	28
	APPENDIX	
	SNAPSHOTS	31

---

**ABSTRACT**

## ABSTRACT

Fractal image compression gives desirable properties like resolution independence, fast decoding, and very competitive rate-distortion curves. It has a very high encoding time, depending on the approach being used.

Fractal compression is asymmetric process taking much longer time to compress and lesser time for decompression. Fractal encoding relies on the fact that all natural, and most artificial, objects contain redundant information in the form of similar, repeating patterns called fractals. This work deals with reduction of encoding time in Fractal image compression. Fractal image compression is a fixed size block-based approach. The encoding process is extremely computationally intensive because of the fact that the searching of the domain pool for the best match is time consuming. The process of matching fractals does not involve looking for exact matches, but instead looking for best fit matches based on the compression parameters.

This work presents a method to reduce the encoding time of this technique by reducing the size of the domain pool based on the Entropy value of each domain block. Analyzing the domain pool, there is a very large set of domain blocks in the pool with high entropy are identified, which are not used in the fractal code. Thus, the search time is reduced by discarding a large fraction of high entropy blocks.



## LIST OF FIGURES

<b>Figure No.</b>	<b>Description</b>	<b>Page No.</b>
4.1	Methods of sampling	12
5.1	Iterative Function System	17
5.2	Fractal Decompression	22

## LIST OF ABBREVIATIONS

**PSNR**-Peak Signal to Noise Ratio

**MSE**-Mean Square Error

**IFS**-Iterative Function System

**JPEG**-Joint Photographic Experts Group

---

## 1.INTRODUCTION

Ever increasing demand for images, sound, video sequences, computer animations and volume visualization, data compression remains a critical issue regarding the cost of data storage and transmission times. While JPEG currently provides the industry standard for still image compression, there is ongoing research in alternative methods. Fractal image compression is one of them. It has generated much interest due to its promise of high compression ratios at good decompression quality and it enjoys the advantage of very fast decompression. Another special feature of fractal image compression is its multi resolution property, i.e. an image can be decoded at higher or lower resolutions than the original, and it is possible to “zoom-in” on sections of the image. These properties made it a very attractive method for applications in multimedia.

Fractal image compression gives high compression ratios of around 1:4 to 1:100 depending upon the fractal content in the image. We can scale any fractal image up or down in size without the introduction of image artefacts or a loss in details that occurs in bitmap images. Despite these features fractal image compression has high (sometimes very high) encoding time.

Several methods have been proposed to overcome this problem. The most common approach for reducing the computational complexity is the classification scheme. In this scheme range and domain blocks are grouped in classes according to their common characteristics. This method reduces the searching space efficiently. However, it required large amount of computations. In clustering methods the domain blocks are classified by clustering their feature vectors in Voronoi cells whose centers are designed from the test image or from a set of training images. For each range block, matches are sought in the neighboring classes only. A different approach is to organize the domain blocks into a tree- structure, which could admit

faster searching over the linear search. Complexity reduction methods that are somewhat different in character are based on reducing the size of the domain pool.

Reducing the encoding time of the fractal compression process is the main objective of this work. In this work a new method to reduce the encoding time of fractal image compression is proposed. This method is based on removing the domain block with entropy,  $S$  from the domain pool. In this way, all the useless similar domains will be removed from the pool achieving a more productive domain too. With lesser, but useful, domains the search time and the mathematical complexity is reduced there by reducing the encoding time.

---

## 2.OVERVIEW

Fractal compression has nothing to do with fractals but it uses the same principles of iteration, and self similarity. The fractal in fractal compression is a (Partitioned) Iterated Function System. A set of linear functions (transformations) when applied to a starting point will generate a new point. When the transformations are repeated on the new points iteratively an image is generated. In this pattern we can produce complex images. Instead of finding transformations that describe the whole picture, find transformations that only apply to portions of a picture, partitioned iterative function system.

## **2.1 FEATURES**

Multi resolution property i.e, the image can be decoded at higher or lower resolutions than the original, and it is possible to zoom-in on sections of the image. Storing images in less memory leads to a direct reduction in storage cost and faster data transmissions. The decoding on the fractal compression is very fast.

## **2.2 CONSTRAINTS**

The long computing in the encoding step still remains the main drawback of this technique. Because good approximations are obtained when many domain blocks are allowed, searching the pool of domain blocks is time consuming. Time taken to encode original image is normally high and that is the disadvantage of fractal image compression.

---

### **3. LITERATURE REVIEW**

Because good approximations are obtained when many domain blocks are allowed, searching the pool of domain blocks is time consuming. In other word, consider an  $N \times N$  image and  $n \times n$  range blocks. The number of range blocks is  $(N/n)^2$ , while the number of the domain blocks is  $(N - 2n + 1)^2$ . The computation of best match between a range block and a domain block is  $O(n^2)$ . Considering  $n$  to be constant, the computation of complexity search is  $O(N^4)$ .

Several methods have been proposed to overcome this problem. The most common approach for reducing the computational complexity is the Classification scheme. In this scheme range and domain blocks are grouped in classes according to their common characteristics. In the encoding phase, only blocks belonging to the same class are compared, thus saving a lot of Computation while keeping the performance in terms of image quality quite close to that of exhaustive search. Jacquin proposed a discrete feature classification scheme based on Ramamurthi and Gersho approach. The domain blocks are classified according to their perceptual geometric features. Only three major types of block are differentiated: shade blocks, edge blocks, and midrange blocks. In the Fisher's classification method, a given image block is divided into four quadrants. For each quadrant, the average and the variance are computed. According to certain combination of these values, 72 classes are constructed. This method reduces the searching space efficiently. However, it required large amount of computations and, the arrangement of these 72 classes is complicated.

In clustering methods the domain blocks are classified by clustering their feature vectors in Voronoi cells whose centres are designed from the test image or from a set of training images. For each range block, matches are sought in the neighbouring classes only. Another discrete feature classification based on mean was proposed by Hurtgen and Stiller. The feature vector is constructed by



comparing the sub-block mean of each quadrant to the block's mean. In this approach the search area for a domain block is restricted to a neighbourhood of the current range. All the above approaches can only reduce the factor of proportionality in  $O(N)$  the time complexity for a search in the domain pool, where  $N$  is the size of the domain pool.

A different approach is to organize the domain blocks into a tree-structure, which could admit faster searching over the linear search. This approach is able to reduce the order of complexity from  $O(N)$  to  $O(\log N)$ . The idea of tree-structured search to speed up encoding has long been used in the related technique of Vector Quantization . Caso et al. and Bani-Eqbal have proposed formulations of tree-search for fractal encoding.

In the feature vector approach introduced by Saupe in a small set of  $d$  real-valued keys is devised for each domain which make up a  $d$ -dimensional feature vector. These keys are carefully constructed such that searching in the domain pool can be restricted to the nearest neighbours of a query point, i.e., the feature vector of the current range. Thus the sequential search in the domain pool is replaced by multi-dimensional nearest neighbour searching, which can be run in logarithmic time. Unfortunately, the feature vector dimension is very high, *i.e.* equal to the number of pixels in the blocks. This limits the performance of this approach as the multi-dimensionality search algorithms. Moreover large amounts of memory are required.

Complexity reduction methods that are somewhat different in character are based on reducing the size of the domain pool. Jacobs et al.'s method uses skipping adjacent domain blocks. Monro localizes the domain pool relative to a given range based on the assumption that domain blocks close to range block are well suited to match the given range block. Saupe's Lean Domain Pool method discards a fraction of domain blocks with the smallest variance.

---

## 4. PROPOSED SYSTEM

The simplest ways to decrease encoding time of this full search problem is to decrease the size of the domain pool in order to decrease the number of domains to be searched. The proposed method reduces the encoding time of fractal image compression by performing fewer searches as opposed to doing a faster search, by excluding many of the domain blocks from the domain pool. This idea is based on the observation that many domains are never used in a typical fractal encoding, and only a fraction of this large domain pool is actually used in the fractal coding.

Analyzing the domain pool, there is a very large set of domain blocks in the pool with high entropy, which are not used in the fractal code. Thus, it is possible to reduce the search time by discarding a large fraction of high entropy blocks, which affect only a few ranges. For these ranges a sub-optimal domain with smaller entropy may be found. In this way, the domain pool is constructed from blocks with the lowest entropy instead of all domains. In this case, the encoding time is heavily reduced by a priori discarding those domains from the pool, which are unlikely to be chosen for the fractal coding. Eq. (6) is used to calculate the entropy value for each domain block. According to this value a decision is taken to determine if this domain can become a part of the domain pool or not. A parameter  $\varepsilon$  will control the domain entropy value in the implementation, with  $\varepsilon$  being a quality parameter since it determines the size of the domain pool.

#### 4.1 ERROR METRICS:

Two of the error metrics used to compare the various image compression techniques are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR). The MSE is the cumulative squared error between the compressed and the original image, whereas PSNR is a measure of the peak error. The mathematical formulae for the two are

$$\text{MSE} = \frac{(R - R1)^2}{\text{NUMBER OF PIXELS}}$$

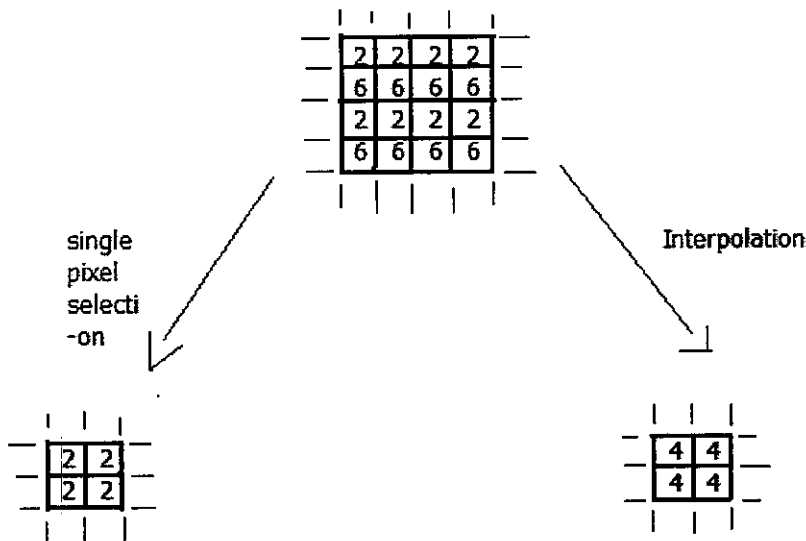
$$\text{PSNR} = 20 * \log_{10} \left( \frac{255}{\sqrt{\text{MSE}}} \right)$$

where R is the original image and R1 is the approximated version of the domain. A lower value of the MSE means lesser error, and as seen from the inverse relation between the MSE and PSNR, this translates to a high value of PSNR. Logically a higher value of PSNR is good because it means that the ratio of signal to noise is higher. Here “signal” is the original image and “noise” is the error in the reconstruction.



## 4.2 MIRRORING, ROTATION AND SCALING:

Image sampling is a fundamental operation of any image processing system. In a digital computer, images are represented as array of finite size numbers. The number of sampling points measured determines the size of the image array. The information at each sampling position in the array represents the average irradiance over the small sampling area and is quantized into finite number of bits. Scaling compresses or expands an image along the coordinate directions.



fig(4.1) Methods of sub-sampling

The figure (4.1) illustrates the two methods of sub-sampling. In the first, one pixel value within a local neighbourhood is chosen (perhaps randomly) to be representative of its surroundings (this method is computationally simple, but can lead to poor results if sampling neighbourhoods are too large). The second method interpolates pixel values within a neighbourhood by taking a statistical sample (such as the mean) of the local intensity values.

The rotation operator performs a geometric transform which maps the position  $(x_1, y_1)$  of a pixel element in an input image onto a position  $(x_2, y_2)$  in an output image by rotating it through a user-specified angle  $\Theta$  about an origin  $O$ . In the most implementations, output locations  $(x_2, y_2)$  which are outside the boundary of the image are ignored. Rotation is not commonly used to improve the visual appearance of an image, although it can be useful as a pre-processor in applications where direction operators are involved.

The rotation operator performs a transformation of the form:

$$X_2 = \cos(\Theta) * (x_1 - x_0) - \sin(\Theta) * (y_1 - y_0) + x_0$$

$$Y_2 = \sin(\Theta) * (x_1 - x_0) + \cos(\Theta) * (y_1 - y_0) + y_0$$

Where  $(x_0, y_0)$  are the coordinates of the centre of rotation (in the input image) and  $\Theta$  is the angle of rotation.

The rotation algorithm, unlike that employed by translation, can produce coordinates  $(x_2, y_2)$  which are not integers. In order to generate the intensity of the pixels at integer position, different heuristics (or re-sampling techniques) may be employed. For example, two common methods include:

- \*Allow the intensity level at each integer pixel position to assume the value of the nearest non-integer neighbour  $(x_2, y_2)$ .

- \*Calculate the intensity level at each integer pixel position based on a weighted average of the  $n$  nearest non-integer value. The weighting is proportional to the distance or pixel overlap of the nearby work stations.

The latter method produces better results but increases the computation time of the algorithm.

### **4.3 SOFRTWARE REQUIREMENTS:**

The hardware and software configurations used to develop this work are given below

#### **4.3.1 HARDWARE CONFIGURATION**

Processor : Pentium IV  
Clock speed : 1.5GHz  
Main memory : 256RAM  
Hard disk : 20GB

#### **4.3.2 SOTWARE CONFIGURATION**

Operating system : Windows XP  
Implementation language : Java  
Tools : jdk1.6.0\_01

---

## **5.IMPLEMENTATION**



## 5.1 PARTITIONING OF THE IMAGE

The image is partitioned into non-overlapping square blocks of size 4x4 called range blocks. The same image is also partitioned into square blocks of size twice that of the range blocks called domain blocks. To find the IFS (Iterative Function System) for the image, for each of the range blocks, all the domain blocks are considered in sequence. Each of these domains is reduced spatially and in the greyscale domain to the size of the range block. Then the MSE between it and its transformation to the range block is chosen and its transformation to the range block is stored.

Typical pixel sizes of the range and domain blocks are

RANGE	4x4	8x8
DOMAIN	8x8	16x16

The domain blocks are double the size of the range blocks. Smaller size of the blocks means a larger compressed file, because of more fractal codes.

## 5.2 ITERATIVE FUNCTION SYSTEM:

The basic idea behind fractal image compression is to express the images as an iterated function system. IFS are a collection of affine maps. Affine maps are combinations of rotations, scaling and transformations. The collage theorem states that if an image can be described by a set of affine maps, the set of maps provides an IFS. The image can then be displayed quickly and zooming will generate infinite levels of fractal details. The problem is how efficiently the IFS for the given image can be generated.

When a set of linear functions (transformations) is applied to a starting point will generate a new point.

$$x_1 = ax + by + e$$

$$y_1 = cx + dy + f$$

When the transformations are repeated on the new points iteratively an image is generated. In this pattern we can produce complex images. Instead of finding transformations that describe the whole picture, find transformations that only apply to portions of a picture, partitioned iterative function system.



Fig(5.1) Examples of Iterated Function System

### 5.3 FRACTAL ENCODING

In the encoding phase of fractal image compression, the image of size  $N \times N$  is first partitioned into non-overlapping range blocks  $R_i$ ,  $\{ R_1, R_2, \dots, R_p \}$  of a predefined size  $B \times B$ . Then, a search codebook (domain pool  $\Omega$ ) is created from the image taking all the square blocks (domain blocks)  $D_j$ ,  $\{ D_1, D_2, \dots, D_q \}$  of size  $2B \times 2B$ , with integer step  $L$  in horizontal or vertical directions. To enlarge the variation, each domain is expanded with the eight basic square block orientations by rotating 90 degrees clockwise the original and the mirror domain block. The range-domain matching process initially consists of a shrinking operation in each domain block that averages its pixel intensities forming a block of size  $B \times B$ .

For a given range  $R_i$ , the encoder must search the domain pool  $\Omega$  for best affine transformation  $w_i$ , which minimizes the distance between the image  $R_i$  and the image  $w_i(D_i)$ , (*i.e.*  $w_i(D_i) \approx R_i$ ). The distance is taken in the luminance dimension not the spatial dimensions. Such a distance can be defined in various ways, but to simplify the computations it is convenient to use the Root Mean Square RMS metric. For a range block with  $n$  pixels, each with intensity  $r_i$  and a decimated domain block with  $n$  pixels, each with intensity  $d_i$ , the objective is to minimize the quality

$$E(R_i, D_i) = \sum_{i=1}^n (sd_i + o - r_i)^2 \quad (1)$$

which occurs when the partial derivatives with respect to  $s$  and  $o$  are zero. Solving the resulting equations will give the best coefficients  $s$  and  $o$ .

$$s = \frac{n \sum_{i=1}^n d_i r_i - \sum_{i=1}^n d_i \sum_{i=1}^n r_i}{n \sum_{i=1}^n d_i^2 - \left( \sum_{i=1}^n d_i \right)^2} \quad (2)$$

$$o = \frac{1}{n} \left( \sum_{i=1}^n r_i - s \sum_{i=1}^n d_i \right) \quad (3)$$

With  $s$  and  $o$  given the square error is

$$E(R_i, D_i)^2 = \frac{1}{n} \left[ \sum_{i=1}^n r_i^2 + s \left( s \sum_{i=1}^n d_i^2 - 2 \sum_{i=1}^n d_i r_i + 2o \sum_{i=1}^n d_i \right) + o \left( on - 2 \sum_{i=1}^n r_i \right) \right] \quad (4)$$

If the denominator in Eq. (2) is zero, then  $s = 0$  and  $o =$

$$\frac{1}{n} \sum_{i=1}^n r_i$$

The parameters that need to be placed in the encoded bit stream are  $s_i$ ,  $o_i$ , index of the best matching domain, and rotation index. The range index  $i$  can be predicted from the decoder if the range blocks are coded sequentially. The coefficient  $s_i$  represents a contrast factor, with  $|s_i| \leq 1.0$ , to make sure that the transformation is contractive in the luminance dimension, while the coefficient  $o_i$  represents brightness offset.

## 5.4 ENTROPY EVALUATION:

Assume that there exists a set of events  $S = \{ x_1, x_2, \dots, x_n \}$ , with the probability of occurrence of each event  $P(x_i) = P_i$ . These probabilities,  $P = \{ p_1, p_2, \dots, p_n \}$ , are such that each  $P_i \geq 0$ , and  $\sum_{i=1}^n P_i = 1$ . The function,

$$I(x_i) = -\log p_i \quad (5)$$

is called the amount of self-information associated with event  $x_i$ . This function is a measure of occurrence of the event  $x_i$ . The function  $I$  focus on one event at a time. In most situations, however, and certainly in the context of data compression, one has to look at the entire set of all possible events to measure content over the entire set. An important concept introduced by Shannon is entropy associated with a set of events, which takes the form:

$$H(p_1, p_2, \dots, p_n) = H(s) = -\sum_{i=1}^n p_i \log p_i \quad (6)$$

Entropy can be defined as the average self-information that is, the mean (expected or average) amount of information for an occurrence of an event  $x_i$ . In the context of coding a message, entropy represents the lower bound on the average number of bits per input value. The function  $H$  has the following lower and the upper limits:

$$0 = H(1, 0, 0, \dots, 0) \leq H(p_1, p_2, \dots, p_n) \leq H\left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right) = \log n \quad (7)$$

In other words, if the events are equally likely, the uncertainty is the highest since the choice of an event is not obvious. If one event has probability 1 and the others probability of 0, the choice is always the same, and all uncertainty disappears.

## 5.5 FRACTAL IMAGE DECODING:

The reconstruction process of the original image consists on the applications of the transformations describe in the fractal code book iteratively to some initial image  $\Omega_{init}$  until the encoded image is retrieved back. The transformation over the whole image can be described as shown below

$$\Omega_1 = \eta(\Omega_{init})$$

$$\Omega_2 = \eta(\Omega_1)$$

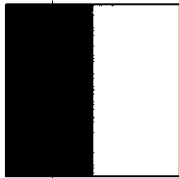
$$\Omega_3 = \eta(\Omega_2)$$

$$\dots = \dots$$

$$\Omega_n = \eta(\Omega_{n-1})$$

$\eta$  can be expressed as two distinct transformations, one for the down sampling and low pass filtering of the image to create the domains for the image, second for the transformations defined by our mapping from the domain blocks in the domain image to the range blocks in the range image as stored in the fractal file. The original image can be obtained from the encoded image in less than 6 iterations.

# Decompression



Initial image

The transformations and mappings are applied on the initial image iteratively until the image is restored.

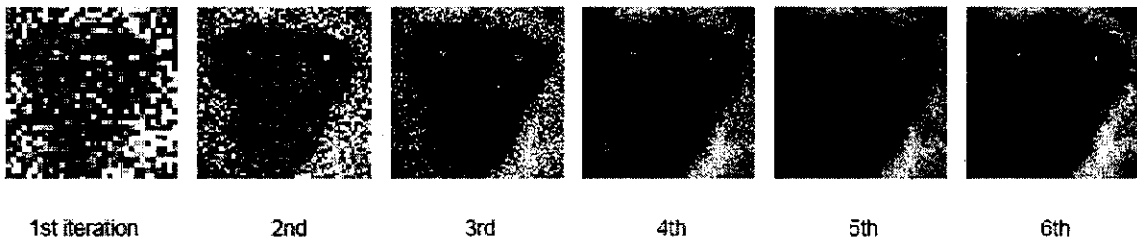


Fig (5.2) Decompression of an image in fractal method

## 5.6 ALGORITHM FOR FINDING THE BEST MATCH:

### Step 1: Initialization (domain pool construction)

```

Choose parameter  $\varepsilon$  ;
Divide the input image into  $N$  domains,  $D_j$ 
For ( $j = 1; j \leq N; j ++$ ) {
    Ent =entropy (  $D_j$  );
    If ( $\text{Ent} \leq \varepsilon$  )
        Push  $D_j$  onto domain pool stack  $\Omega$  }

```

### Step 2: Choose a tolerance levels $I_c$ ;

### Step 3: Search for best matches between range and domain blocks

```

For (  $i = 1 ; i \leq \text{num\_range} ; i ++$  ) {
    min_error =  $I_c$  ;
    For (  $j = 1 ; j \leq \text{num\_domain}; j ++$  ) {
        Compute  $s, o$ ;
        If ( $0 \leq s < 1.0$ )
            If (  $E( R_i, D_j ) < \text{min\_error}$  ) {
                min_error =  $E( R_i, D_j )$ ;
                best_domain[ $i$ ] =  $j$  ; }
    }
    If ( $\text{min\_error} == I_c$  )
        Set  $R_i$  uncovered and partition it into 4 smaller blocks;
    Else
        Save_coefficients(best_domain,  $s, o$ );
}

```



---

**6.CONCLUSION**

## CONCLUSION

The implementation of fast fractal image compression based on entropy has proven to be successful. The proposed system proves to be very efficient for colour images. The interface has been made simple so that the user can easily compress the images by fractal compression and also by fast fractal compression.

The work was very challenging to work with. This work presented a lot of hidden difficulties in terms of logic as well as implementation. This work proved to be mentally stimulating and intellectually satisfying. It is shown that the encoding time is reduced considerably in the proposed system.

---

## 7.FUTURE ENHANCEMENTS

## **FUTURE ENHANCEMENTS**

This technique is developed in such a way that any enhancements may be made with ease. In future the proposed method can be improved by increasing the speed of hybrid coders (gaining better results than JPEG) that are based on fractal coders and transform coders so as to improve their performance.

The encoding time may be reduced further using some newer techniques in future. The compression is not lossless and it results in some loss of pixels in the image. This may be overcome by using some other techniques.

---

**8.REFERENCES**

## REFERENCES

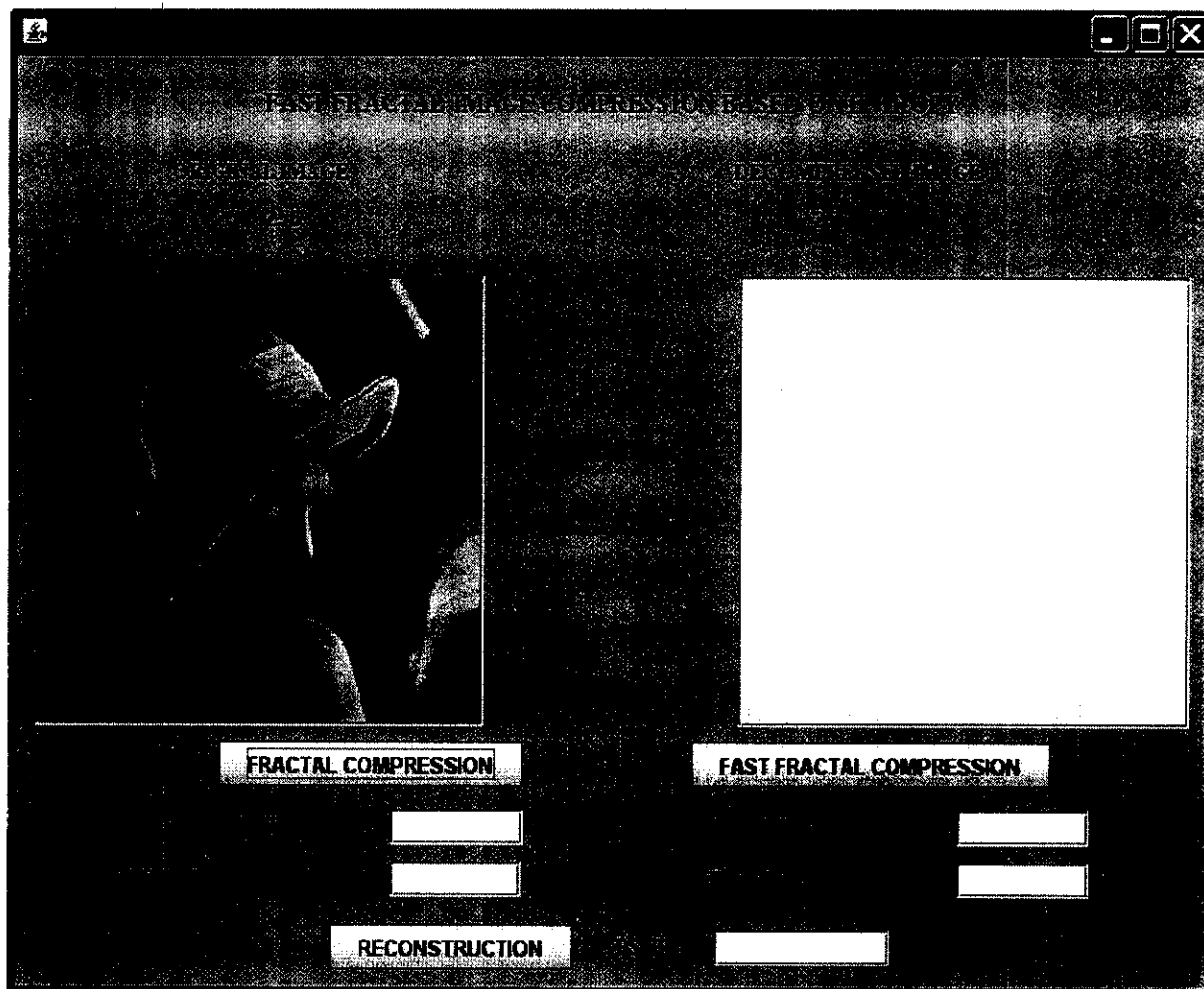
- [1] Barnsley, M. and Jacquin, A.' Applications of Recurrent Iterated Function Systems to Images . SPIE Visual Communications and Image Processing, Vol. 1001.
- [2] Jacquin, A. E.' Image Coding Based on a Fractal Theory of Iterated Contractive Image Transform'. IEEE trans. on Image Processing, Vol. 1, N
- [3] Barnsley, M. and Hurd, L.' Fractal Image Compression on Image Processing: Mathematical Methods and Applications'. pp. 183-210, Clarendon Press, Oxford, 1997.
- [4] Ramanurthi, B. and Gersho, A.' Classified Vector Quantization of Image'. IEEE Trans. Communication, COM-34, Vol.11, pp. 1105-1115, 1986.
- [5] Fisher, Y.' Fractal Image Compression: Theory and Applications'. Springer-Verlag, New York, 1994.
- [6] Hamzaoui, R.' Codebook Clustering by Self-Organizing Maps for Fractal Image Compression'. In NATO ASI Conf. Fractal Image Encoding and Analysis, Trondheim, July 1995. Fractals, Vol. 5, Supplementary issue, April 1997.
- [7] Hamzaoui, R. and Saupe, D.' Combining Fractal Image Compression and Vector Quantization'. IEEE Trans. on Image Processing, 9(2), pp.197-208, 2000.
- [8] Hurtgen, B. and Stiller, C.' Fast Hierarchical Codebook Search for Fractal Coding of Still Images'. Proceeding of SPIE, Vol. 1977, pp. 397-408, 1993.
- [9] Po, L.M. and Chan, C.K.' Adaptive Dimensionality Reduction Techniques for Tree-Structured Vector Quantization'. IEEE *Trans.* on Communications, Vol. 42, No. 6, pp. 2246-2257, June 1994.
- [10] Caso, G., Obrador , P. and Kuo, C.-C.' Fast Methods for Fractal Image Encoding'. Proc. SPIE Visual Communication and Image Processing \_ 95, Vol. 2501, pp.583-594, 1995.
- [11] Bani-Eqbal, B.' Enhancing the Speed of Fractal Image Compression'. Optical Engineering, Vol. 34, No. 6, pp.1705-1710, June 1995.

- [12] Saupe, D.' Accelerating Fractal Image Compression by Multi-dimensional Nearest Neighbor Search'. In Proc. Data compression Conference, March 28-30, 1995.
- [13] Saupe, D.' Fractal Image Compression via Nearest Neighbor Searching'. In Conf. Proc. NATO ASI, Fractal Image Coding and Analysis Trondheim, July 1995.
- [14] Tong, C.S. and Man, W.' Adaptive Approximation Nearest Neighbor Search for Fractal Image Compression'. IEEE Trans. on Image Processing, 11(6), pp.605-615, 2002.
- [15] Jacobs, E.W., Fisher, Y. and Boss, R.D. 'Image Compression: A study of the Iterated Transform Method'. Signal Process, Vol. 29, pp. 251-263, 1992.
- [16] Monro, D.M. and Dudbridge, F.' Approximation of Image Blocks'. in Proc. Int. Conf. Acoustics, Speed, Signal Processing, Vol. 3, pp.4585-4588, 1992.
- [17] Saupe, D.' Lean Domain Pools for Fractal Image Compression'. Proceedings IS&T/SPIE 1996 Symposium on Electronic Imaging: Science & Technology Still Image Compression II, Vol. 2669, June 1996.
- [18] Polvere, M. and Nappi, M. 'Speed-Up in Fractal Image Coding: Comparison of Methods'. IEEE Trans. on Image Processing, Vol. 9, No. 6, pp.1002-1009, June 2000.
- [19] Wohlberg, B. and Jager, G.' A review of the Fractal Image Compression Literature'. IEEE Trans. On Image Processing, Vol. 8, No. 12, pp. 1716-1729, Dec. 1999.
- [20] Saupe, D. and Hamzaoui, R.' Complexity Reduction Methods for Fractal Image Compression'. In I.M.A. Conf. Proc. on Image Processing; Mathematical methods and applications, Sept., J.M. Blackedge (ed.), 1994.
- [21] Gharavi-Alkhansari, X. and Huang, T.S.' Fractal Image Coding Using Rate-Distortion Optimized matching Pursuit'. Proc. SPIE, pp. 265-304, 1996.
- [22] Tong, C.S. and Pi, M.' Fast Fractal Encoding Based on Adaptive Search'. IEEE Trans. on Image Proc.10, No.9, pp.1269-1277, Sept. 2001.

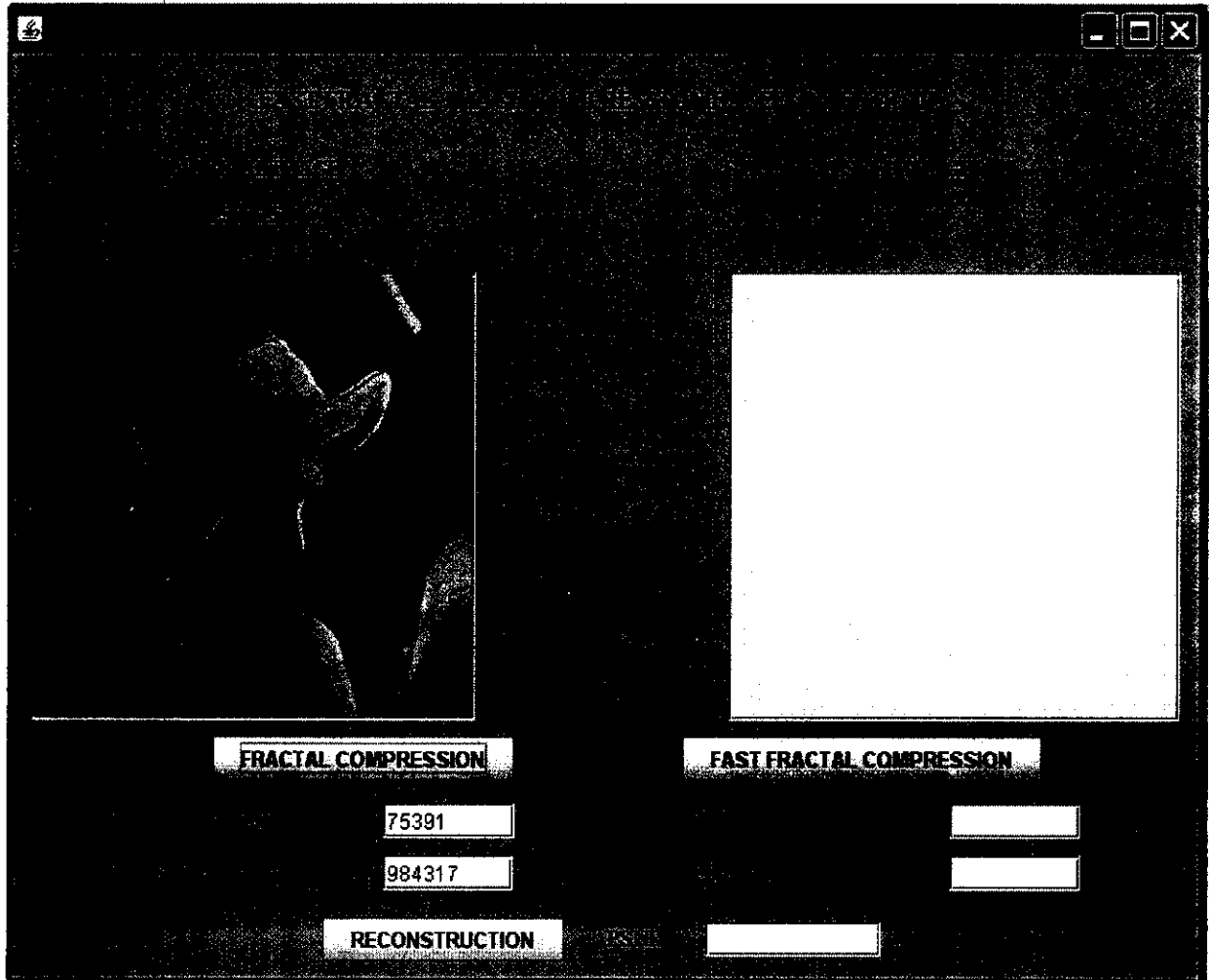
---

**SNAPSHOTS**

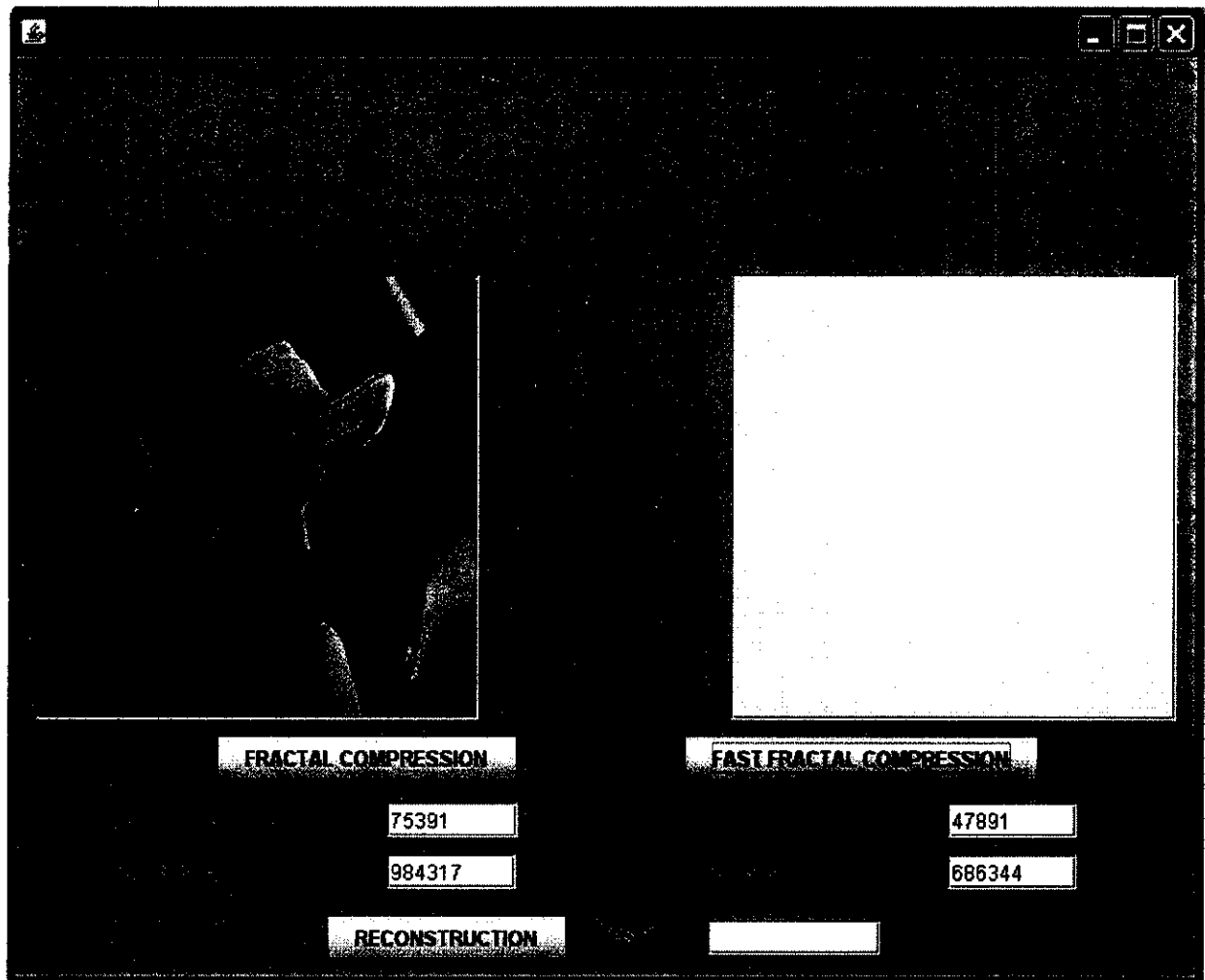




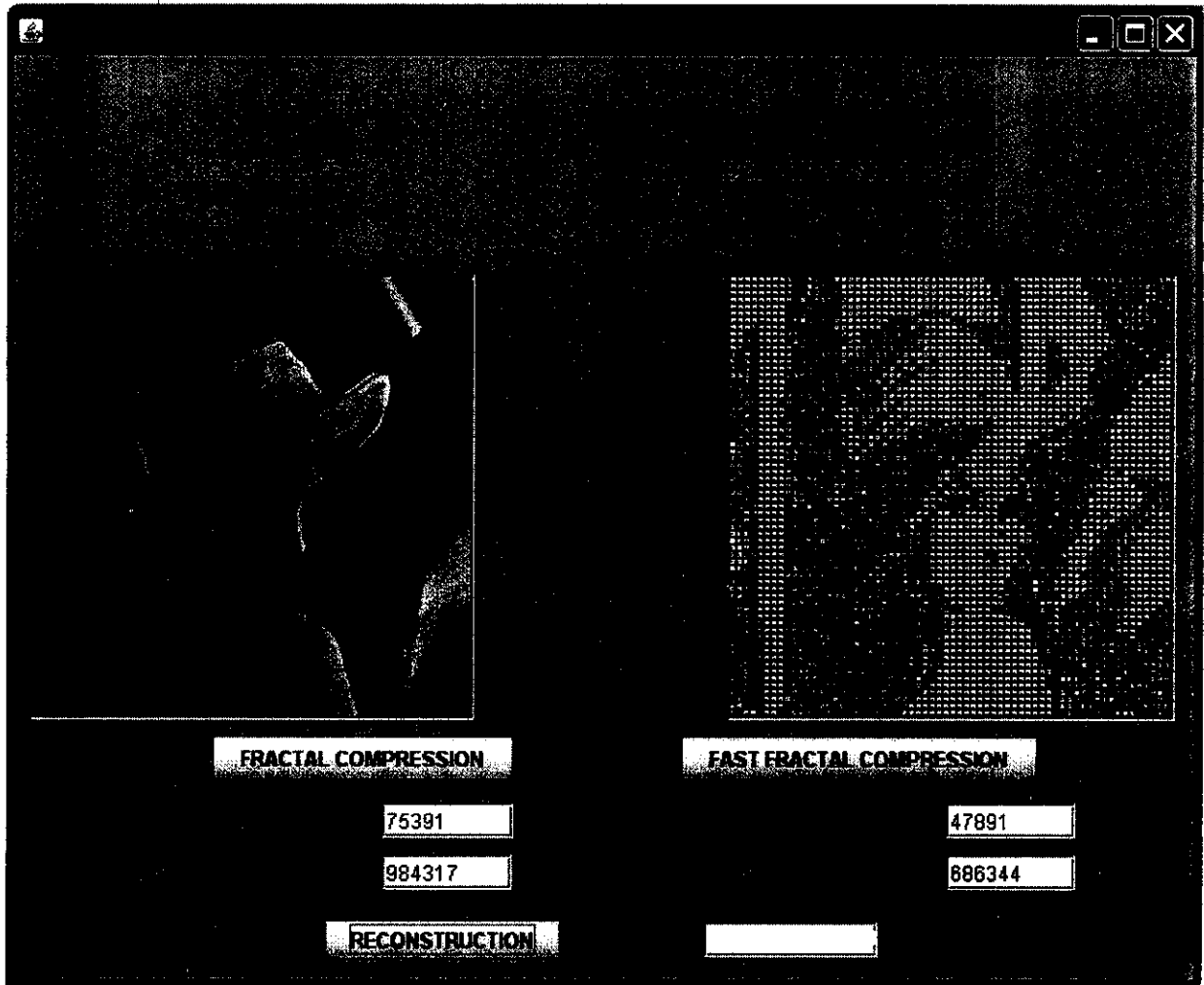
**Graphical User Interface**-Screen with the input image (lena).



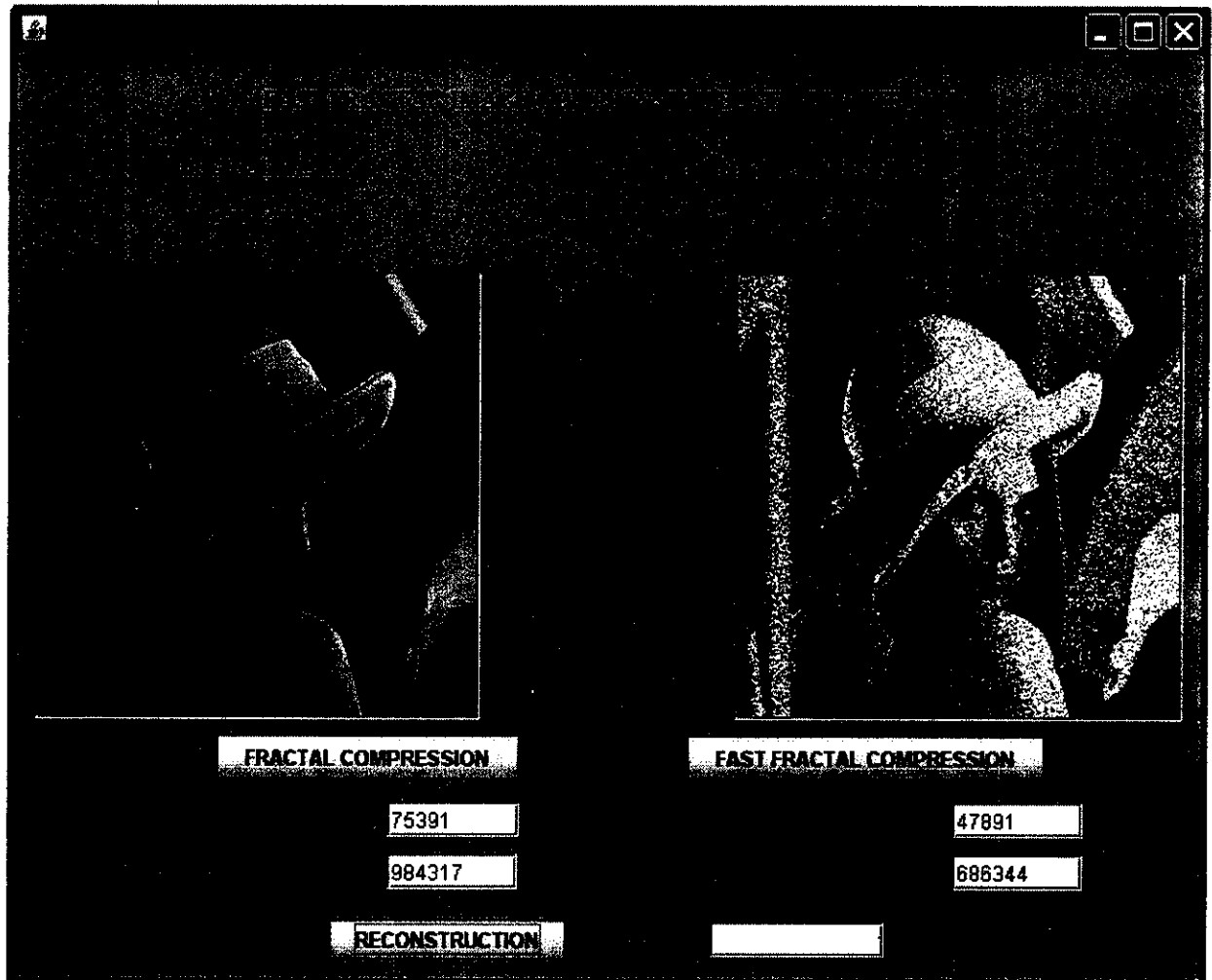
**Fractal compression**-the image is compressed using the basic fractal compression. The encoding time and the file size are calculated.



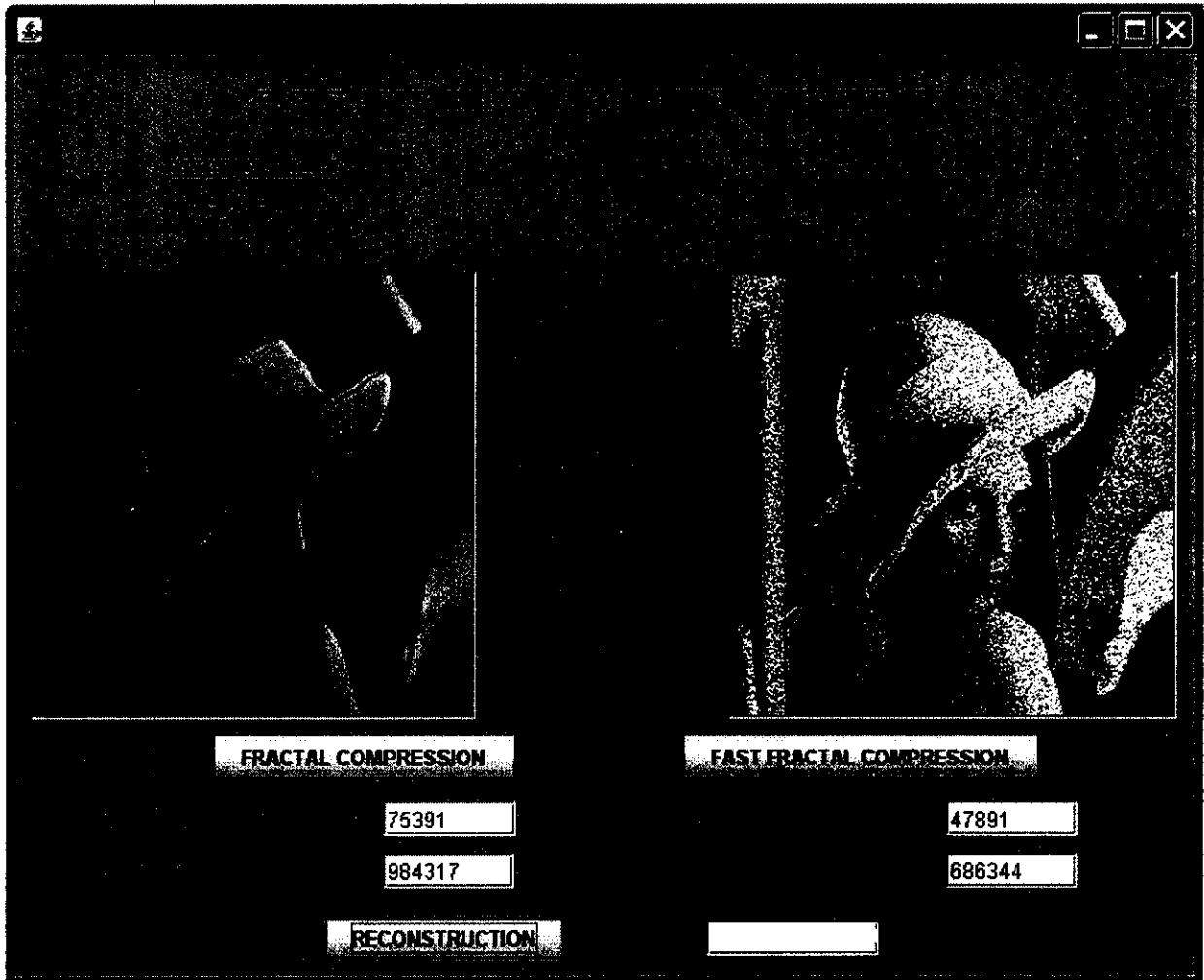
**Fast Fractal Compression**-The same image is compressed using the proposed entropy based method. The encoding time and file size is calculated.



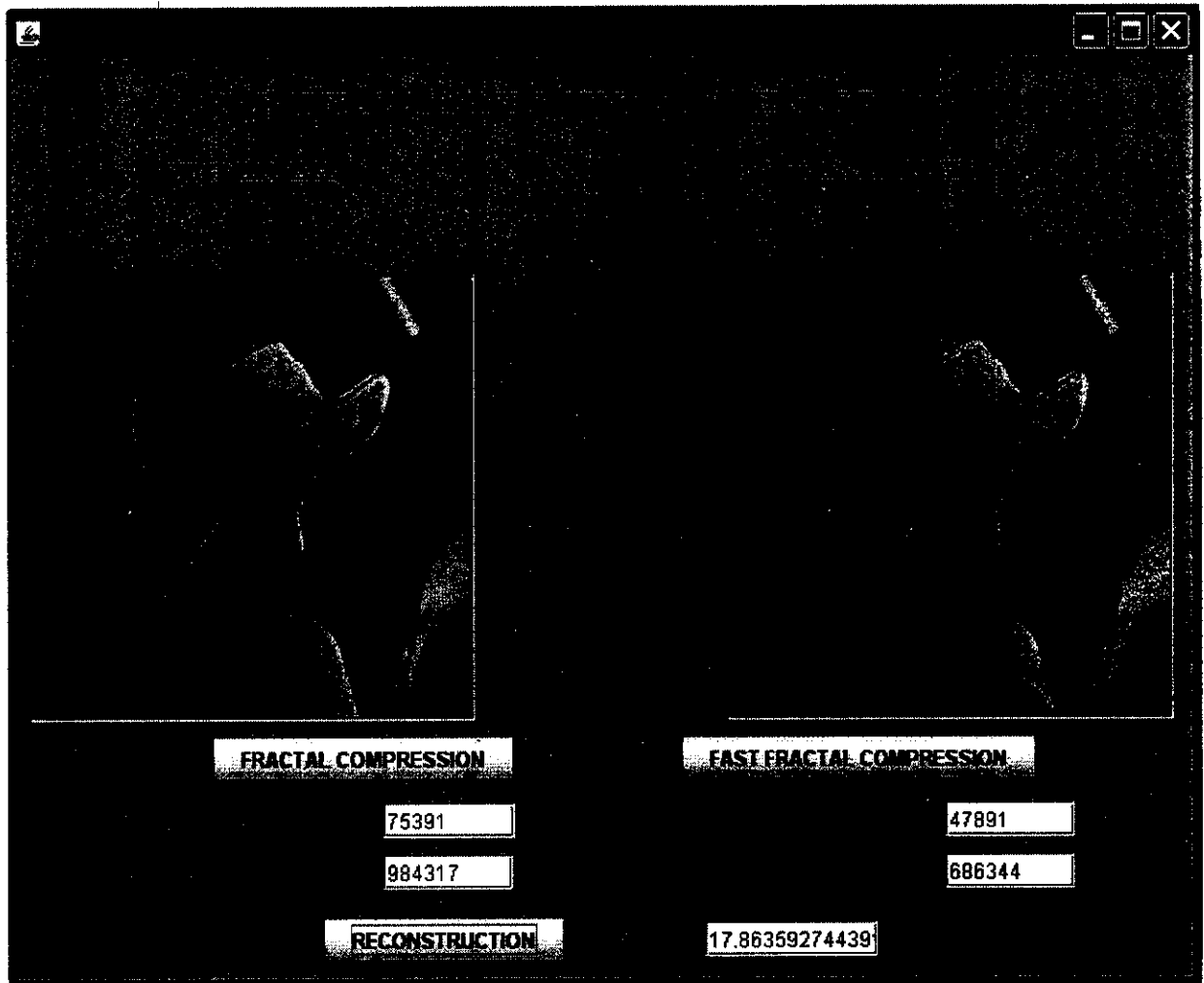
**Reconstruction-Screen with reconstructed image after the first iteration.**



**Reconstruction**-Screen with reconstructed image after the second iteration.



**Reconstruction**-Screen with reconstructed image after the third iteration.



**Reconstruction**-Screen with reconstructed final image after the fourth iteration and the result of calculation of the PSNR value.