Fractal Image Compression with Adaptive Quadtree Partitioning and Lossless Encoding on the Parameters of Affine Transformations

Utpal Nandi and Jyotsna Kumar Mandal

Abstract This paper proposes two ways to improve the compression ratio achieved by Fractal image compression. During subdivision of the input image into range and domain blocks, Adaptive Quadtree Partitioning is used. During the final save, the parameters of the affine transforms of the fractal compressed images are losslessly compressed using two methods—once by Modified Region Based Huffman (MRBH) coding, and another time by its variant, MRBHM. The proposed techniques offer much better compression ratios most of the time, keeping PSNRs unchanged. However, compression time of the proposed techniques are significantly more than their counterparts.

Keywords Fractal compression · AQP · MRBH · MRBHM · IFS · PIFS

1 Introduction

If we look at the texture of the floor of a room, we will observe that there are many repeating patterns. If we take a copy of a small area of the floor, we will find that there are many areas which are nearly identical to that copy. And if we rotate it, scale it, mirror it, we will find that we can get even more areas to match. If we now

U. Nandi (✉)
Vidyasagar Univesity, Midnapore, India
e-mail: nandi.3utpal@gmail.com

J.K. Mandal
University of Kalyani, Kalyani, India
e-mail: jkm.cse@gmail.com

J.K. Mandal et al. (eds.), *Information Systems Design and Intelligent Applications*,
Advances in Intelligent Systems and Computing 340,
DOI 10.1007/978-81-322-2247-7_9

store the positions of the copy and the matching area of the floor, including the rotations etc. that we just made, we can create a mathematical representation of that part of the floor. We can do this until we have described the entire floor. This is what fractal compression does.

The main idea in fractal image compression is to look for parts in an image that resembles other parts in the same image [1]. To do this, the image is divided into *range blocks* (small) and *domain blocks* (big), and a domain block is used to describe a range block. For each range block, we search for that particular domain block that most closely resembles the range block. Transformations such as scaling, translation, rotation, shearing, etc. and adjustment of brightness/contrast are used on the domain block in order to get the best match.

For partitioning an image into domain and/or range blocks, generally the quadtree partitioning is used. It involves dividing a rectangular region into four equal rectangular regions, simply by joining the mid-points of opposite sides.

But the problem of quadtree partitioning is that the scheme does not use context of the image during partitioning, so some self-similar structures get divided by the partition. This problem is solved by the Adaptive Quadtree Partitioning (AQP) scheme [2] that is discussed in Sect. 2. In this paper, the performance of Fractal Image Compression with Adaptive Quadtree Partitioning (FICAQP) is enhanced by applying loss-less encoding on the parameters of the affine maps. The lossless techniques used are MRBH [3] and MRBHM [4] that are discussed in Sect. 3. The proposed techniques are explained in detail in Sect. 4. The results are discussed in Sect. 5. The conclusions are made in Sect. 6, and the references follow.

2 Fractal Image Compression Using Adaptive Quadtree Partitioning

The Adaptive Quadtree Partitioning scheme partitions the image blocks in content-dependent way. So self-similar structures remain in the same block and do not get partitioned. The detailed algorithm of the scheme is as follows:

Algorithm 1 Adaptive Quadtree Partitioning Algorithm

Require: A block of pixels spanning R rows $\times$ C columns where each pixel value is $P_{i,j}$, $(0 \leq i < R,\ 0 \leq j < C)$. The top-left point of this range is (p, q) (Fig.1)

1: **Partition the block horizontally into two sub-blocks** $REC1$ **and** $REC2$ **along the row** R_p**.** (Fig.2)

1.1: For each row i, calculate the sum of pixel values:

$$S_i = \sum_{j=q}^{q+C-1} P_{i,j} \quad \text{where} \quad p \leq i \leq p + R - 1. \tag{1}$$

1.2: For each row i, calculate the absolute value of successive differences between pixel value sums of i^{th} and $(i+1)^{th}$ rows:

$$D_i = |S_i - S_{i+1}| \quad \text{where} \quad p \leq i \leq p + R - 1. \tag{2}$$

1.3: Apply a linear biasing function $min(i, R - i - 1)$ to each D_i to get biased horizontal differences between i^{th} and $(i+1)^{th}$ rows:

$$H_i = \min(i - p, R - i + p - 1) \cdot D_i \quad \text{where} \quad p \leq i \leq p + R - 1. \tag{3}$$

1.4: Calculate the partitioning row R_p as:

$$R_p = \max(H_i) \quad \text{subject to} \quad p \leq i \leq p + R - 1. \tag{4}$$

2: **Partition the upper sub-block** $REC1$ **vertically into two sub-sub-blocks** $REC1.1$ **and** $REC1.2$ **along the column** C_{p1} (Fig.3)

2.1: For each column j, calculate the sum of pixel values:

$$S'_j = \sum_{i=p}^{R_p-1} P_{i,j} \quad \text{where} \quad q \leq j \leq q + C - 1. \tag{5}$$

2.2: For each column j, calculate the absolute value of successive differences between pixel value sums of j^{th} and $(j+1)^{th}$ columns:

$$D'_j = \left|S'_j - S'_{j+1}\right| \quad \text{where} \quad q \leq j \leq q + C - 1. \tag{6}$$

2.3: Apply a linear biasing function $min(j, C - j - 1)$ to each D'_j to get biased horizontal differences between j^{th} and $(j+1)^{th}$ columns:

$$H'_j = \min(j - q, C - j + q - 1) \cdot D'_j \quad \text{where} \quad q \leq j \leq q + C - 1. \tag{7}$$

2.4: Calculate the partitioning column C_{p1} as:

$$C_{p1} = \max(H'_j) \quad \text{subject to} \quad q \leq j \leq q + C - 1. \tag{8}$$

3: **Partition the lower sub-block** $REC2$ **vertically into two sub-sub-blocks** $REC2.1$ **and** $REC2.2$ **along the column** C_{p2} (Fig.4)

3.1: For each column j, calculate the sum of pixel values:

$$S''_j = \sum_{i=R_p}^{R-1} P_{i,j} \quad \text{where} \quad q \leq j \leq q + C - 1. \tag{9}$$

3.2: For each column j, calculate the absolute value of successive differences between pixel value sums of j^{th} and $(j+1)^{th}$ columns:

$$D''_j = \left| S''_j - S''_{j+1} \right| \quad \text{where} \quad q \leq j \leq q + C - 1. \tag{10}$$

3.3: Apply a linear biasing function $min(j, C-j-1)$ to each D''_j to get biased horizontal differences between j^{th} and $(j+1)^{th}$ columns:

$$H''_j = \min(j - q, C - j + q - 1) \cdot D''_j \quad \text{where} \quad q \leq j \leq q + C - 1. \tag{11}$$

3.4: Calculate the partitioning column C_{p2} as:

$$C_{p2} = \max(H''_j) \quad \text{subject to} \quad q \leq j \leq q + C - 1. \tag{12}$$

3 RBH, MRBH and MRBHM Coding

Classical Huffman Coding [5] is based on the frequency of symbols appearing in the input string. To reduce storage space, the most frequent symbol of the whole string is assigned with the shortest code.

The Region Based Huffman (RBH) [3] Coding is a modification of the classical Huffman Coding. Here, the input string is divided into a number of regions (say *N*). Huffman codes are first calculated for each symbol with respect to the entire string. Then the codes are re-calculated taking each region separately. Now, if the code of the most frequent element of a particular region is longer than the code of the most frequent element of the entire string, their codes are interchanged (i.e. the regional-most-frequent-element gets the shorter code). This interchanging information is attached to the output. The elements of the region are now encoded with this changed code. Otherwise, the original codes are used to encode the symbols. This technique is repeated for all regions.

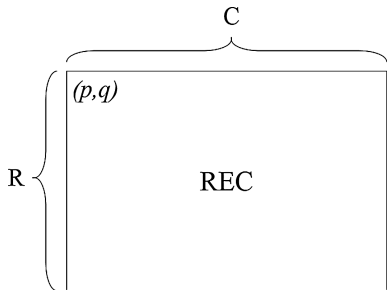

Fig. 1 Adaptive quadtree partitioning (AQP): initial image

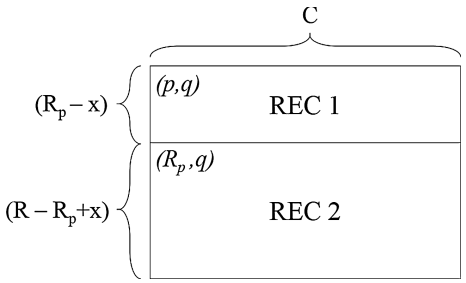

Fig. 2 AQP: after step 1

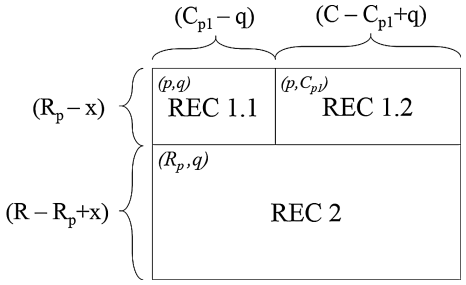

Fig. 3 AQP: after step 2

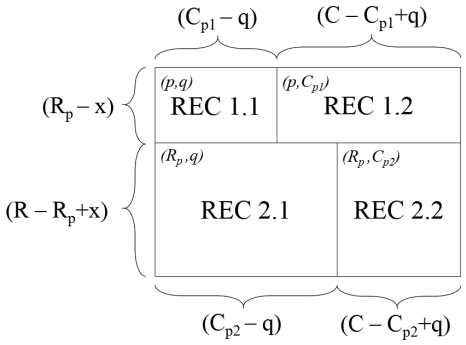

Fig. 4 AQP: after step 3

RBH coding relies on optimum selection of the number of regions (N) that the input string will be partitioned into. If the value of N is not properly chosen, the compression is not so effective.

The Modified Region Based Huffman (MRBH) Coding [3] uses a Region Selection Algorithm (RSA) [3] to select an optimum value of N for the RBH algorithm to be effective. For a given range of values of $N \in (L1, L2)$, the algorithm calculates the number of bits that can be reduced in the RBH coding of a string, by using code interchange. It returns that value of N which reduces the maximum number of bits.

Modified Region Based Huffman coding with Multiple Interchange of code [4], abbreviated as MRBHM, is a further improvement of the MRBH coding. Here, Huffman code of more than one high frequency symbol of a region is interchanged with the codes of the high frequency symbols of the entire input string. A Modified Region Selection Algorithm (MRSA) [4] is used to select the optimum value of N (number of regions).

4 Proposed Techniques

The proposed techniques try to enhance the performance of Fractal Image Compression using Adaptive Quadtree Partitioning (FICAQP), by applying lossless encoding on the parameters of the affine maps of fractal compressed images. The technique is basically a two-step method as shown in Fig. 5.

In the first step the input image is compressed by fractal technique using Adaptive Quadtree Partitioning scheme. The compression technique begins from the whole image, and continues recursively until the domains and ranges are either within some specified RMS tolerance, or the ranges are smaller than a specified minimum size. It produces a collection of affine maps as shown in Eq. (13). The constants $a_{i,j}$ represent the scaling factors, while the constants $d_{i,j}$ represent the top-left corner of the domains. Two constants—c_i and b_i—representing the contrast and brightness factors respectively, specify the luminance part. Therefore, an affine map is basically a collection of parameters.

Fig. 5 Compression technique

Fig. 6 Decompression technique

In the second step, a lossless encoding is used to compress the parameters of the affine maps. Here, MRBH coding is used. The proposed technique is termed as Fractal Image Compression with Adaptive Quadtree Partitioning and MRBH coding on the parameters of affine transforms (FICAQP-MRBH).

During decompression, first the compressed affine maps are decompressed using MRBH coding. Then these affine maps generate the image using Fractal Image Decompression with Adaptive Quadtree Partitioning as shown in Fig. 6.

Then, a variant of the above technique is also proposed which is termed as Fractal Image Compression with Adaptive Quadtree Partitioning and MRBHM coding on the parameters of affine transformations (FICAQP-MRBMH). In this proposed FICAQP-MRBHM, instead of using MRBH coding, MRBHM coding is used in the second step of the compression technique. The first step is identical to the previous proposed technique.

$$W_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_{i,1} & a_{i,2} & 0 \\ a_{i,3} & a_{i,4} & 0 \\ 0 & 0 & c_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} d_{i,1} \\ d_{i,2} \\ b_i \end{bmatrix} \tag{13}$$

5 Results and Discussion

For experimental purpose, five grey scale image files with dimensions (200×320) and size 64,000 Bytes have been taken. The "GS" suffix is for "non-formatted" grey-scale image files. The techniques proposed in this paper viz. FICAQP-MRBH and FICAQP-MRBHM are compared with basic Fractal Image Compression with Quadtree Partitioning (FICQP) technique and Fractal Image Compression with Adaptive Quadtree Partitioning (FICAQP) technique.

The comparison of space savings for the five image files have been made in Table 1. The comparison of PSNRs for the same are given in Table 2 while that of compression time are in Table 3. The graphical comparison of space savings, PSNRs and compression times are illustrated in Figs. 7, 8 and 9 respectively.

Table 1 Comparison of space savings (%) of images

Image	FICQP	FICAQP	FICAQP-MRBH	FICAQP-MRBHM
CHEETA.GS	82.56	82.76	84.91	85.07
LISAW.GS	82.59	82.91	84.87	84.96
ROSE.GS	87.45	87.50	89.03	89.42
MOUSE.GS	85.87	85.93	87.84	87.95
CLOWN.GS	82.82	83.13	84.52	84.74

Table 2 Comparison of PSNRs of images (in dB)

Image	FICQP	FICAQP	FICAQP-MRBH	FICAQP-MRBHM
CHEETA.GS	27.50	27.93	27.93	27.93
LISAW.GS	38.91	41.24	41.24	41.24
ROSE.GS	28.97	29.25	29.25	29.25
MOUSE.GS	38.03	38.67	38.67	38.67
CLOWN.GS	29.20	29.72	29.72	29.72

Table 3 Comparison of compression time of images

Image	FICQP	FICAQP	FICAQP-MRBH	FICAQP-MRBHM
CHEETA.GS	140	310	350	360
LISAW.GS	170	360	390	400
ROSE.GS	130	290	310	330
MOUSE.GS	150	340	370	380
CLOWN.GS	190	420	440	450

The proposed two techniques offer comparatively better space savings than their counterparts, for all five images. Between the two proposed techniques, FICAQP-MRBHM is slightly better than FICAQP-MRBH in terms of space savings. The PSNRs of the final images are the same compared to existing FICAQP techniques. However the compression times taken by the proposed techniques are much more than existing techniques for all five images. FICAQP-MRBHM takes much more compression time than others.

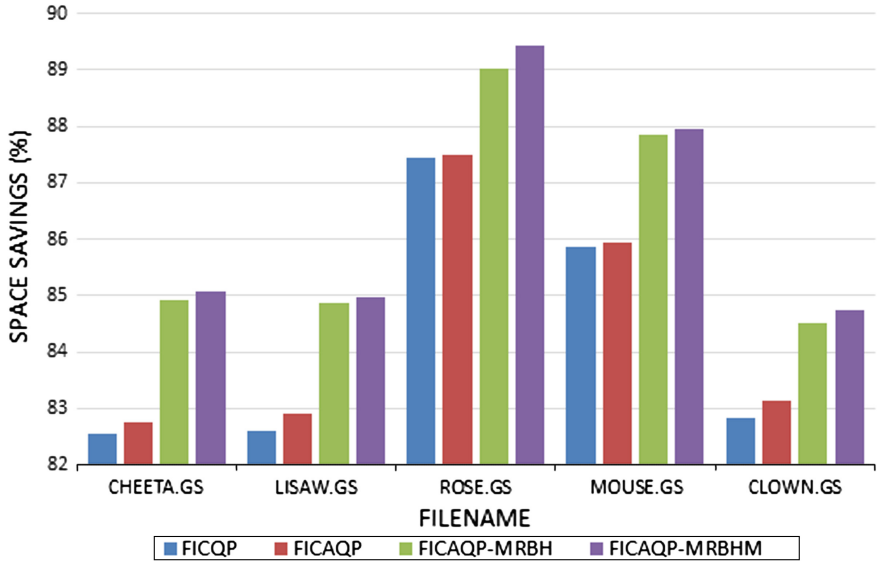

Fig. 7 Graphical comparison of space savings (%)

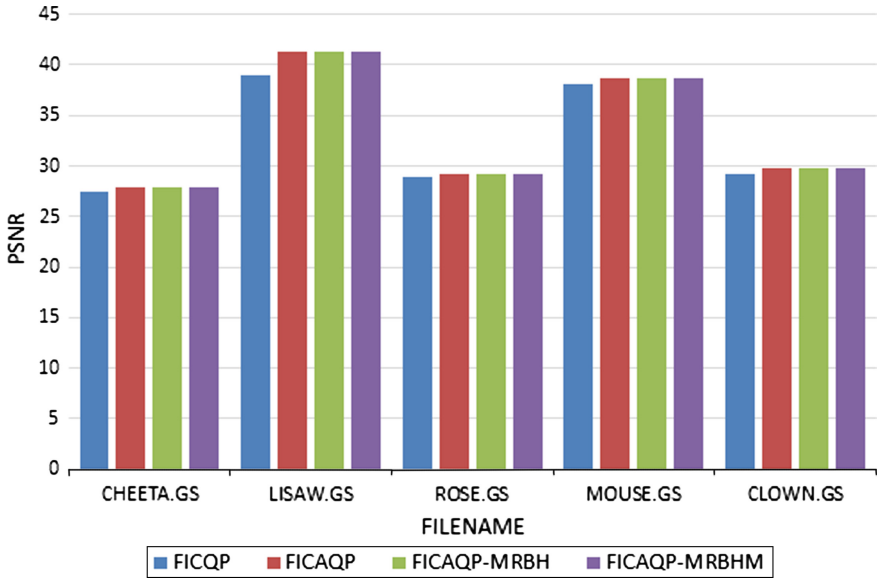

Fig. 8 Graphical comparison of PSNRs (in dB)

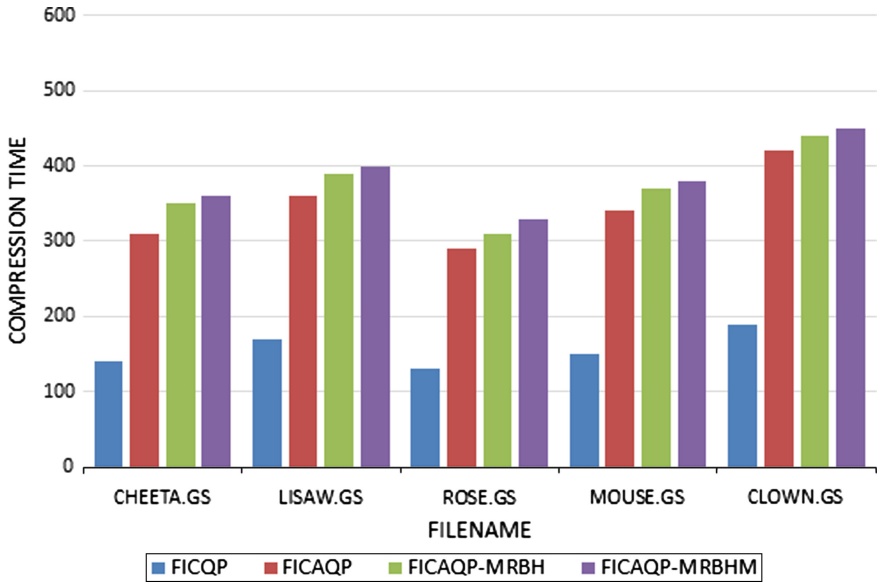

Fig. 9 Graphical representation of compression times of images

6 Conclusion

The proposed Fractal Image Compression by using Adaptive Quadtree Partitioning and loss-less encoding techniques MRBH and MRBHM coding on the parameters of affine transform FICAQP-MRBH and FICAQP-MRBHM improve the compression rates of images significantly. Though the PSNR of existing FICAQP and proposed techniques are same, the encoding and decoding time of images are increased. Among the two proposed techniques, FICAQP-MRBHM offers much better compression rates. But it takes more compression time than its counterparts. To reduce compression time, other fast lossless coding techniques can be applied.

Acknowledgment The authors would like to thank CST Department, University of Kalyani and CS Department, Bangabasi College, for providing the infrastructures.

References

1. Fisher, Y. (ed.): Fractal Image Compression: Theory and Application. Springer, New York (1994)
2. Nandi, U., Mandal, J.: Fractal image compression with adaptive quadtree partitioning scheme. In: International Conference on Signal, Image Processing and Pattern Recognition (SIPP-2013), vol. 3, pp. 289–296 (2013)

3. Nandi, U., Mandal, J.K.: Region based huffman (RBH) compression technique with code interchange. Malays. J. Comput. Sci. **23**(2), 111–120 (2010)
4. Nandi, U., Mandal, J.: Region based huffman compression with region wise multiple interchanging of codes. Adv. Model. Simul. Tech. Enterprises (AMSE), France **17**(2), 44–58 (2012)
5. Huffman, D.A., et al.: A method for the construction of minimum redundancy codes. Proc. IRE **40**(9), 1098–1101 (1952)