Enhanced Fractal Image Compression using Haar Wavelet

C. S. Tong[*] and K. P. Cheuk

Department of Mathematics, Hong Kong Baptist University

Kowloon Tong, Hong Kong

Email: cstong@hkbu.edu.hk

Abstract

Fractal image compression offers a high compression ratio with extremely short decompression time. However, it suffered from long compression time. In this paper, a new block classification method based on the Haar wavelet transform is introduced. When used in conjunction with our domain block pool reduction scheme, the enhanced method can improve the speed of compression over 50 times with minimal loss of decompression accuracy.

Keywords: Fractal Image Compression, Haar Wavelet, Enhancing Compression Speed, Domain Block Pool Reduction

[*] Corresponding author: Tel: +852 2339 7023; fax: +852 2336 1505; email: cstong@hkbu.edu.hk

1. Introduction

As computers become more and more powerful, the temptation to use digital images become irresistible. Along with this comes the serious issue of storing and transferring the huge volume of data that represent digital images. Fractal Image Compression is one of many techniques that can store images more efficiently. Based on the properties of fractal images [1], Fractal Image Compression was first introduced by Michael F. Barnsley [2]. Using the Iterated Function System (IFS) algorithm [3-4], it can achieve a high compression ratio with a short decompression time. However, the compression time is typically extremely long. Some of the methods used to decrease the encoding time are reviewed in [5-7] and it is also possible to enhance the decoding time [8]. With the help of Haar Wavelet Transform and a domain block pool reduction scheme, we will show in this paper that we can increase the speed of compression by up to 50 times with only a small drop in the Signal-to-Noise Ratio (SNR) of the decompressed image.

2. The Iterated Function Systems

An IFS [9-10] is a collection of contractive affine transformations $\{\omega_i : \mathbb{R}^2 \to \mathbb{R}^2 \mid i = 1, \cdots, n\}$ which maps the plane $\mathbb{R}^2$ to itself. This collection of transformations defines a map *W*:

$$W(\bullet) = \bigcup_{i=1}^{n} \omega_i(\bullet).$$

Two conditions are usually imposed on the contractive affine transformations ω_i, namely, (i) they preserve shapes, and (ii) they satisfy the Open Set Condition (i.e. ω_i maps disjoint open sets to disjoint open sets).

The Contractive Mapping Fixed Point Theorem [10] states that for every contractive map *W* on a space of images, then there is a unique special image, called the *attractor* and denoted x_w, which is a fixed point of *W*, i.e.

$$W(x_w) = x_w.$$

Moreover, given any arbitrary input image I_0,

$$x_w = \lim_{n\to\infty} W^n(I_0) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots (*)$$

The problem of Fractal Image Compression (FIC) can now be formulated as follows. Given an image I to be encoded, we wish to find a corresponding contractive map W for which I is its fixed point. Then the image I can be represented (or coded) by the parameters which define the contractive map. To decode and recover the original image, we use equation (*) which tells us that repeated iteration of the contractive map on any starting image will converge to its fixed point, i.e. converge to *I*, as required. The convergence, hence the decompression, is usually very fast and is indeed one of the advantage of FIC.

In practice, it may not be possible to find the exact map *W*. Fortunately the Collage Theorem [10] ensures that a good approximation for *W* will have a fixed point that is close to the fixed point of *W*. This gives us the flexibility to trade off compression accuracy with compression time and compression ratio by choosing the level of approximation for *W*.

3. Basic Fractal Image Compression (BFIC)

The basic fractal image compression algorithm as described in [5] will serve as the benchmark in this paper. First of all, we form a pool of $2n \times 2n$ domain blocks by sliding a $2n \times 2n$ window around the image pixel by pixel, each such block can be indexed by the coordinates of its top left pixel. For a $N \times N$ image, this would generate a large domain block pool of size $(N-n+1)^2$. Each domain block is now spatially contracted, denoted $\zeta(\bullet)$, to $n \times n$ blocks by replacing each disjoint 2×2 sub-blocks by their neighbourhood averages. This collection of spatially contracted $n \times n$ blocks forms a virtual codebook that will be used to generate the contractive map.

The key concept of fractal image compression is self-similarity. In order to satisfy the Open Set Condition, the input image is partitioned into disjoint $n \times n$ blocks called range blocks, and for each range block we find the corresponding contractive affine transformation for it by searching through the virtual codebook generated above. Specifically, we identify the affine transformed, spatially contracted domain block which best match the given range block.

To satisfy the shape-preserving condition, we shall restrict the geometric part of the affine transformations to 8 isometries, denoted $\rho(\bullet)$, which represent rotating the $n \times n$ blocks by multiples of 90 degrees, and/or flipping them about the horizontal and vertical axes [5]. Lastly, we apply so called massic transformations, σ, to the image intensities $f(i, j)$ of the pixels in the form:

$$\sigma(f(i, j)) = \alpha \ f(i, j) + \beta ,$$

where α controls the contrast scaling, and β controls the luminance shift. These parameters are usually quantized to increase compression ratios [5].

Thus the contractive map W we sought is a collection of affine transformations of the form:

$$\omega_i = \sigma \circ \rho \circ \zeta \ .$$

4. Haar Wavelet Transformation

The 1-D mother Haar wavelet (see Figure 1) is defined as:

$$\varphi(t) = \begin{cases} 1, \text{for } 0 < t < \frac{1}{2}, \\ -1, \text{for } \frac{1}{2} < t < 1, \\ 0, \text{otherwise.} \end{cases}$$

In 2-D, the Haar wavelet transform can be summarized in the scheme below:

1. Given an input block. Take the mean of the intensity of the block to obtain the first wavelet coefficient e_1.

2. Divide the block into four equal sub-blocks. Take the mean values of each sub-blocks and subtract these sub-block means from the overall mean (i.e. of the input block before the division) to obtain the next 4 coefficients, e_2 to e_5.

3. Repeat the procedure in step 2 for each sub-block (see Figure 2) until we reach the pixel level.

Note that some of the coefficients can be derived from the others so that they are not all independent. We shall make use of some of these coefficients to classify the range and domain blocks.

5. Classified-Blocks Fractal Image Compression Algorithm (CFIC)

In the Basic Fractal Image Compression Algorithm (BFIC), we have a lot of domain blocks in the domain block pool and the search through such a large pool for each and every range block constitutes the bulk of the computation cost in compression. Clearly, we could substantially reduce the search time if we can classify the blocks into different types and restrict the search to blocks of the same type. An example of block classification uses a measure of edge content in each block for classification [5]. It achieves a reasonable speed up, but the technique is difficult to fine-tune and the measure of edge content is arbitrary.

In this paper, we note that Haar wavelet coefficients provide information on the distribution of relative intensities and can be used to classify blocks. The first coefficient describes the overall mean of the block and is not particularly useful as the massic transformation part of the affine transformation can take care of such differences between blocks. However, the next four coefficients, e_2 to e_5, are very useful. By sorting these four coefficients, we effectively rank the four sub-blocks in intensities. Since there are 4! = 24 permutations, we may classify the blocks into 24 types and substantially reduce the pool size for each search within blocks of the same type. Furthermore, as in [5], blocks which are essentially uniform should be treated in a class of its own, and these are identified when all the coefficients e_2 to e_5 are equal (or to within a tolerance level).

Furthermore, we can classify blocks within each type by using the next level of Haar wavelet coefficients to build up a tree-structure classification scheme, somewhat similar to the quadtree fractal compression scheme proposed in [10]. This may be particularly useful when large range block sizes are used as in some parent-child multi-level fractal compression schemes [5].

6. Domain Pool Reduction

To further enhance the speed of compression, we now consider reducing the size of the domain pool further by skipping blocks that differ by just one pixel-wide columns or rows. This reduces the pool to a quarter of its size and thus is expected to lead to an extra speed up factor of about 4. On the face of it, this reduction of search time is only achieved at the expense of reducing the spatial resolution. However, this needs not be the case since all domain blocks are spatially contracted and this has the effect of converting the loss of spatial resolution to a loss of intensity resolution.

This effect is demonstrated in Figure 3 where two edge blocks with gray level transition of Δ are spatially contracted. Edge blocks A and B differed in one pixel-column, but after contraction, both became the same spatially-sized edge-blocks, except that in the case of block B, the gray level transition dropped from Δ to $\Delta/2$. But this change in intensity can be exactly compensated for by a massic transformation!

For general blocks, it may not be possible to exactly compensate for the loss of intensity resolution after spatial contraction, but with an appropriate choice of massic transformation, we could minimize such loss which is present in any case because we need to quantize massic transformation coefficients for high compression anyway. We thus conclude that our proposed scheme for domain block pool reduction is justified. Fractal Image Compression using both the block classification scheme discussed in section 5 and the domain block pool reduction scheme proposed here will henceforth be referred to as the Enhanced Fractal Image Compression (EFIC) scheme.

7. Results and Conclusions

All the three compression schemes discussed above, namely, BFIC, CFIC, and EFIC, have been implemented on a 486 PC, with the customary choice of $n = 4$. The compression results on the 256×256 Lena image are summarized in Table 4. The decompressed images are shown in Figure 5, and visually they are practically identical to the original image. The BFIC method took over 50 hours to compress, but the CFIC method was faster by a factor of 14, whilst the EFIC method is faster still by a factor of 4, as expected. In all three methods, the decompressions were very fast, with convergence by about the tenth iterations. The signal-to-noise ratio (SNR) of the decompressed images for all three methods are comparable, with only a slight drop from 33.9780 dB for the BFIC method to 32.6556 dB for the EFIC method. The compression ratios are just under 5 in these experiments. They could be improved by changing block sizes and adjusting the quantizations for the transformation parameters, but that is outside the scope of this paper.

We further tested the methods on 5 different 256×256 images (see Figure 6) and the results are summarized in Table 7. The speed up of the CFIC over BFIC varied from 6.5 to 16.5. But the extra speed up in going from CFIC to EFIC was around 4 in all 5 cases, just as we expected. Moreover the SNR for the EFIC images remain very close to the corresponding CFIC images, differing by at most 1 dB. These results confirmed our justification for the Domain Block Pool Reduction scheme.

The CFIC images were all nearly identical to those generated by the BFIC, with SNR differing by less than 0.4 dB. For images A & E, the CFIC decompressed images were actually slightly superior to the BFIC in terms of the SNR, with an increase of 0.05 dB for image A and 0.01 for image E. This very minor improvement was probably due to the slight discrepancies in the convergence during decompressions. These results clearly demonstrated that our block classification scheme is robust and can lead to substantial speed up in compression with virtually no change in compression quality.

Comparing the EFIC method to the BFIC method, the compression time has been speeded up by about 50 times, with the exception of image A which showed a speed up factor of over 25. Since the EFIC images were very close to the CFIC images which in turn were virtually identical to the BFIC images, the EFIC method has thus gained very good speed up in compression at a very minor loss of around 1 dB in SNR.

To conclude, we find that the CFIC method has achieved quite impressive speed up in compression time over the BFIC method with virtually no difference in the quality of compression. The adoption of our Domain Block Pool Reduction scheme has yielded an extra speed up factor of 4 at the expense of a minor drop of compression quality of around 1 dB which is normally unnoticeable visually. Moreover, our compression schemes, both the CFIC and the EFIC, can be used in conjunction with other speed-enhancing techniques such as the classification scheme based on edge-content suggested by Jacquin [5] and the tree-arrangement of domain blocks proposed by Bani-Eqbal [6].

References

[1] M. F. Barnsley, “Fractal Everywhere”, Academic Press, 1993.

[2] M. F. Bransley and A.D. Sloan, “A Better Way to Compress Images”, Byte, pp. 215-223, January 1988.

[3] J. C. Hart, “Fractal Image Compression and Recurrent Iterated Function Systems”, *IEEE Computer Graphics and Applications*, pp. 25-33, 1996.

[4] E. W. Jacobs, Y. Fisher, R. D. Boss, “Image Compression: a study of the iterated transform method”, *Signal Process*. **29**, pp. 251-263, 1992.

[5] A. Jacquin. “Fractal image coding based on a theory of iterated contractive image transformations”. SPIE **Vol. 1360** Visual Communications and Image Processing ’90, pp. 227-237, 1990.

[6] B. Bani-Eqbal, “Enchancing the speed of fractal image compression”, *Optical Engineering*, **Vol. 34**, No. 6, pp. 1705-1710, June 1995.

[7] D. R. McGregor, R. J. Fryer, P. Cockshott and P. Murray, “Faster Fractal Compression”, Dr. Dobb’s Journal, pp. 34-40, January 1996.

[8] Sarah B. M. Bell, “Fractals: a fast, accurate and illuminating algorithm”, *Image and Vision Computing*, **Vol. 13**, No. 4, pp. 253-257, May 1995.

[9] M. F. Barnsley and L. P. Hurd, “Fractal Image Compression”, AK Peter Ltd., 1993.

[10] Y. Fisher, “Fractal Image Compression: Theory and Application”, Springer-Verlag New York, 1995.

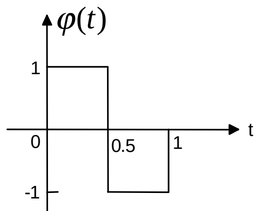


Figure 1

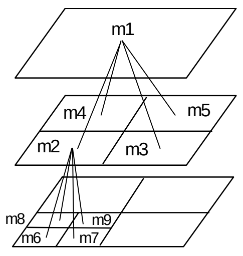


Figure 2

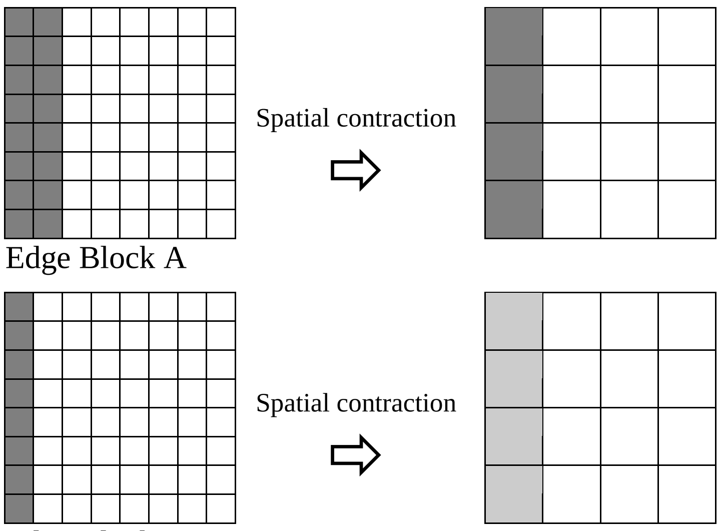


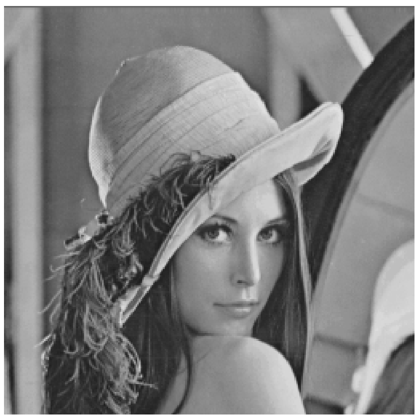
Figure 3: Effect of Spatial Contraction on Edge Blocks

	BFIC	CFIC	EFIC
Compression Time (mins)	3204	227	56
Speed Up	1	14	57
SNR 10 iterations	33.9780	33.6783	32.6556
Drop in SNR	N/A	0.2997	1.3224
Compression Ratio	4.57	4.57	4.92

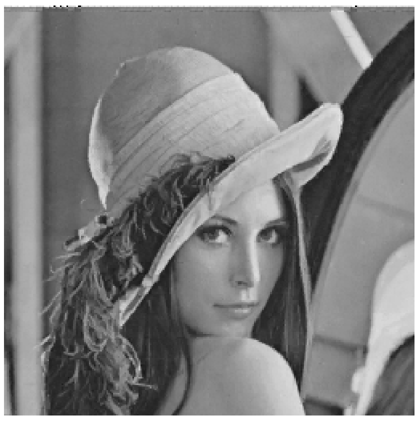
Table 4 Compression Results for the Lena image

(a) Original Lena Image

(b) BFIC compression



(c) CFIC compression



(d) EFIC compression

Figure 5 Compressed images of Lena

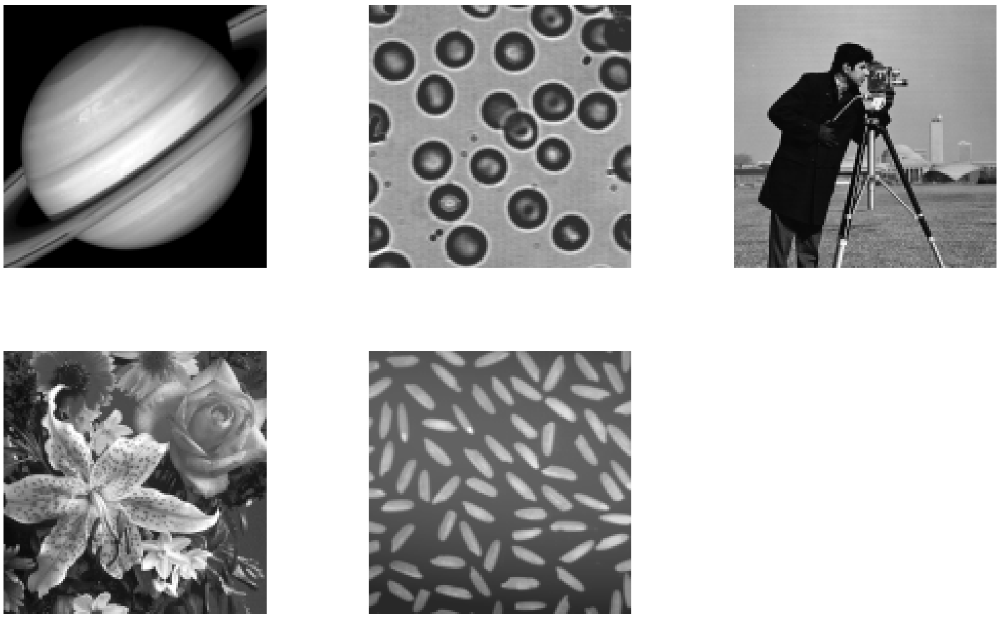


Figure 6: Images A, B, C on Top Row, D & E on Bottom Row

	BFIC		CFIC		EFIC	
Images	SNR	Compression Time	SNR	Speed up	SNR	Speed up
A	43.5420	54h 50m	43.5907	6.5	43.3863	25.5
B	33.7764	54h 50m	33.7303	16.5	32.7665	56.7
C	31.6121	54h 20m	31.2761	14.7	30.5356	56.2
D	32.5601	54h 22m	32.3141	14.9	31.3087	58.3
E	40.1402	54h 10m	40.1525	12.5	39.6352	48.5

Table 7 Compression Results for Images in Figure 6