ELSEVIER

Signal Processing 63 (1997) 263–268

SIGNAL PROCESSING

An FFT-based technique for fast fractal image compression

M. Ramkumar, G.V. Anand*

Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India

Received 13 January 1997; received in revised form 8 September 1997

Abstract

We present a new FFT-based technique of fractal image compression which can significantly speed up the process of matching domain and range blocks. In this method the isometric transformations of a domain block are chosen to permit block matching through cross correlation of range and domain blocks, which is implemented through the FFT. A suboptimal choice of domain blocks by performing a magnitude Fourier domain comparison of the range and domain blocks prior to correlation further speeds up the block matching process. © 1997 Elsevier Science B.V. All rights reserved.

Zusammenfassung

Wir stellen eine neue FFT-gestützte Technik zur fraktalen Bildkompression vor, die den Anpassungsprozeß für die Eingangs- und Ausgangsblöcke erheblich beschleunigt. Bei dieser Methode werden die isometrischen Transformationen eines Eingangsblocks so gewählt, daß eine Blockanpassung durch Kreuzkorrelation von Eingangs- und Ausgangsblöcken ermöglicht wird, die durch die FFT implementiert wird. Durch die Durchführung eines Vergleichs im Fourrierbetragsbereich der Eingangs- und Ausgangsblöcke vor der Korrelation wird eine suboptimale Auswahl der Eingangsblöcke ermöglicht, die den Blockanpassungsprozeß noch weiter beschleunigt. © 1997 Elsevier Science B.V. All rights reserved.

Résumé

Nous présentons une technique nouvelle basée sur la FFT pour la compression fractale d'images qui peut accélérer de maniére significative le processus d'appariement des blocs domaines avec les blocs cibles. Dans cette méthode les transformations isométriques d'un bloc domaine sont choisies de façon à permettre un appariement de bloc par inter-corrélation des blocs domaines avec les blocs cibles, implantée via la FFT. Un choix sous-optimal des blocs domaines par comparaison des amplitudes dans le domaine Fourier avant corrélation accélère encore le processus d'appariement de blocs. © 1997 Elsevier Science B.V. All rights reserved.

Keywords: Fractal image compression; Block matching; FFT-based technique

* Corresponding author. Tel.: +91 80 309 2277; e-mail: anandgv@ece.iisc.ernet.in.

1. Introduction

In fractal or IFS image compression, the image to be coded is completely tiled by non-overlapping 'range' blocks, $\boldsymbol{R}_i, i = 1, \ldots N_r$, and possibly overlapping 'domain' blocks $\boldsymbol{D}_k, k = 1, \ldots, N_d$ [4]. We then approximate every range block $\boldsymbol{R}_i$ by

$$\hat{\boldsymbol{R}}_i = s_i T_i(\boldsymbol{D}_k) + o_i, \quad i = 1, \ldots, N_r, \tag{1}$$

where $\boldsymbol{D}_k$ may be any one of the N_d domain blocks. The parameters s_i and o_i are usually called the scale factor and the offset, respectively. T_i is a transformation that involves spatial contraction of the domain blocks to the size of the range block, followed by an isometric operation (reshuffling of pixels inside the decimated domain block). The encoding process, therefore, involves a computationally expensive search for the best fitting domain block $\boldsymbol{D}_k$ for each range block $\boldsymbol{R}_i$. A lot of research activity has been directed at methods to reduce the computational complexity of the block matching process [5]. In this paper, a new FFT-based technique of fast fractal image compression is presented. The new technique offers a significant reduction in computational complexity for the cost of a slight reduction in the SNR of the decoded image. Another method based on FFT was proposed recently by Saupe and Hartenstein [6], but it does not offer any significant computational advantages over the conventional block matching technique.

2. Error minimisation

Normally, the range blocks $\boldsymbol{R}_i$ and the domain blocks $\boldsymbol{D}_k$ are square matrices of pixel values, of size $r \times r$ and $nr \times nr$, respectively, where the integer $n \geqslant 2$. Contraction or decimation of a domain block is achieved by partitioning it into r^2 sub-blocks of size $n \times n$ and assigning a single pixel value to each sub-block equal to the average of the pixel values within it. A set of t blocks is generated from each contracted domain block through isometric transformations. Normally eight isometric transformations ($t = 8$) consisting of four reflections and four rotations are chosen [4]. Thus an enlarged pool of transformed blocks, called codebook blocks $C_j, j = 1, \ldots, N_d t$, is generated from the original pool of domain blocks $\boldsymbol{D}_k, k = 1, \ldots, N_d$. Each range block $\boldsymbol{R}_i$ is then approximated by an affine transformation of a codebook block

$$\hat{\boldsymbol{R}}_i = s_i \boldsymbol{C}_j + o_i, \quad i = 1, \ldots, N_r, \tag{2}$$

the choice of the matching codebook block $\boldsymbol{C}_j$ and the parameters s_i and o_i being made so as to minimise the approximation error $d(\boldsymbol{R}_i, \hat{\boldsymbol{R}}_i)$. The metric d is usually the squared error. Error minimisation is usually achieved using least squares regression [3]. A simpler procedure based on the inner product of the range and codebook blocks is described below. Let each range block be ordered as a vector by scanning. Let $\boldsymbol{r}_i$ be the zero mean range vector corresponding to the range block $\boldsymbol{R}_i$, obtained by subtracting the mean value of $\boldsymbol{R}_i$ from all the elements. Let $\boldsymbol{c}_j$ denote the zero-mean codebook vector obtained similarly from the codebook block $\boldsymbol{C}_j$. Both $\boldsymbol{r}_i$ and $\boldsymbol{c}_j$ are N-dimensional vectors, where $N = r^2$. The problem at hand is to choose, for a given i, the codebook vector $\boldsymbol{c}_j$ and the scalars s_i^j and o_i^j so as to minimise the error

$$\begin{aligned}\varepsilon_i^j = d(\hat{\boldsymbol{r}}_i, \boldsymbol{r}_i) &= d(s_i^j \boldsymbol{c}_j + o_i^j, \boldsymbol{r}_i) \\ &= \langle \boldsymbol{r}_i - s_i^j \boldsymbol{c}_j - o_i^j, \boldsymbol{r}_i - s_i^j \boldsymbol{c}_j - o_i^j \rangle.\end{aligned} \tag{3}$$

The symbol $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors. Since both $\boldsymbol{r}_i$ and $\boldsymbol{c}_j$ are zero-mean vectors, it is obvious that $o_i^j = 0$. It can also be easily seen that for minimising ε_i^j,

$$s_i^j = \frac{\gamma_{ij}}{\sigma_{c_j}^2}, \tag{4}$$

where

$$\gamma_{ij} = \langle \boldsymbol{r}_i, \boldsymbol{c}_j \rangle, \quad \sigma_{c_j} = \langle \boldsymbol{c}_j, \boldsymbol{c}_j \rangle. \tag{5}$$

The error is now given by

$$\varepsilon_i^j = \left\langle r_i - \frac{\gamma_{ij}}{\sigma_{c_j}^2} c_j, r_i - \frac{\gamma_{ij}}{\sigma_{c_j}^2} c_j \right\rangle = \sigma_{r_i}^2 - \frac{\gamma_{ij}^2}{\sigma_{c_j}^2}, \tag{6}$$

where $\sigma_{r_i}^2 = \langle \boldsymbol{r}_i, \boldsymbol{r}_i \rangle$. Eq. (6) gives the minimum approximation error of $\boldsymbol{r}_i$ for a given c_j. For minimising ε_i^j over all $\boldsymbol{c}_j$, we have to find the codebook block for which $\gamma_{ij}^2/\sigma_{c_j}^2$ is maximum. If we normalise all codebook blocks by dividing them by σ_{c_j}, we just have to find the maximum of $|\langle \boldsymbol{r}_i, \tilde{\boldsymbol{c}}_j \rangle|$, where $\tilde{\boldsymbol{c}}_j = \boldsymbol{c}_j / \sigma_{c_j}$.

We shall apply this error minimisation technique to the FFT-based block matching technique described below.

3. FFT-based block matching

Having seen that block matching involves just one inner product of the range and codebook vectors, we now present a method which reduces the computational complexity to less than N multiplications per comparison on an average. This is done by choosing the isometric transformations of the decimated domain blocks to be different from the usual eight (4 rotations and 4 reflections) [1]. We convert each decimated domain block into a vector using a suitable scanning procedure. We choose a set of $N = r^2$ transformations of each decimated domain block by inverse scanning all possible circular shifts of the corresponding vector to obtain N codebook blocks for each domain block. Now the inner product of a range block with each member of a set of N codebook blocks derived from the same domain block is obtained by circular correlation of the range vector and the parent shrunken domain vector. This would involve N^2 multiplications. However, we can implement the circular correlation using the FFT algorithm to reduce the number of multiplications to the order of $N \log_2 N$. Thus, the average number of multiplications per comparison is reduced from N to $\log_2 N$. Note that the number of isometric transformations used in this method is much more than the usual 8. For example, for 8×8 range blocks, we permit 64 transformations. We can also use the flipped or reverse scan vector whose Fourier transform is just the complex conjugate of the Fourier transform of the original scan vector to yield 64 more transformations. In general for range block size $r \times r$, we permit $2r^2$ transformations of each shrunken domain block. As the number of transformations for each domain block is increased we can hope to reduce the number of domain blocks (to keep the number of codebook blocks constant) *provided all transformations are 'useful'*.

If $\boldsymbol{r} \in \Re^N$ is a range vector and $\boldsymbol{d} \in \Re^N$ is a shrunken domain vector (assume that the means are subtracted from $\boldsymbol{r}$ and $\boldsymbol{d}$ and further that $\boldsymbol{d}$ is normalised to have unit norm), the inner products of $\boldsymbol{r}$ with all circular shift transformations of $\boldsymbol{d}$ can be written as a vector $\boldsymbol{p} \in \Re^N$:

$$\boldsymbol{p} = \mathscr{F}^{-1}(\mathscr{F}(d) \cdot \mathscr{F}^*(r)), \tag{7}$$

where $\mathscr{F}$ stands for DFT and $\mathscr{F}^{-1}$ stands for inverse DFT, and a dot between two vectors denotes their Hadamard product (the vector whose elements are obtained by multiplying the corresponding elements of the multiplicands). The inner products of $\boldsymbol{r}$ with all the circular shift transformations of the flipped vector $\boldsymbol{d}_r$ obtained from $\boldsymbol{d}$ can be represented as

$$\boldsymbol{p}_r = \mathscr{F}^{-1}(\mathscr{F}(d) \cdot \mathscr{F}(r)). \tag{8}$$

Let $\tilde{\boldsymbol{p}}$ denote a vector of $2N$ elements, obtained by appending the elements of $\boldsymbol{p}_r$ to those of $\boldsymbol{p}$. The best suited transformation is given by the index of the element of $\tilde{\boldsymbol{p}}$ with the largest magnitude.

4. Implementation

The above scheme of FFT-based block matching was tried on a variety of 512×512 images. Among the $N!$ possible ways of scanning a block of N pixels, many were tried out, but only two of them were found to be useful in terms of the SNR of the decoded image. The first one is the normal scanning method of stacking the rows of a block to form a single row vector. The second method is the Hilbert scan ordering of pixels of a block [2]. The Hilbert scanning of a 4×4 block is shown in Fig. 1.

The 512×512 image was divided into 4096 range blocks of size 8×8 and 256 non-overlapping domain blocks of size 32×32. As explained in the previous section, 128 transformations of each shrunken domain block were permitted, making the effective number of code book blocks equal to 32 768. The scaling down of the domain blocks to the size of the range blocks was done by averaging sixteen pixels to one. Comparison of the range and domain blocks was done as indicated by Eqs. (7) and (8). The IFS code for each range block consists of

- the scale factor (5 bits),
- mean of the range block (7 bits),

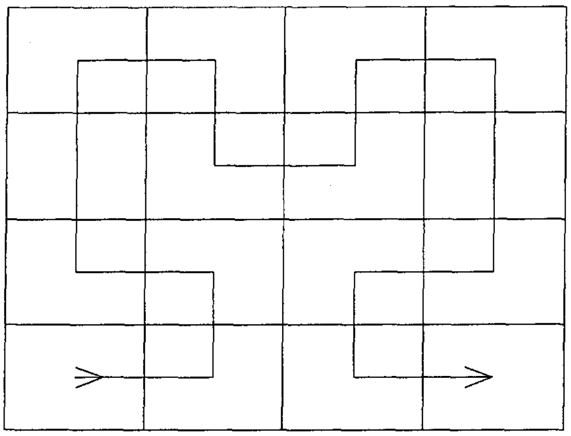
Fig. 1. Hilbert curve scanning of a 4 × 4 block.

- address of one of the 256 domain blocks (8 bits), and
- transformation used (7 bits).

The results obtained by this method are compared with those obtained by the conventional (Jacquin's) technique using 3969 domain blocks of size 16 × 16 (domain blocks overlapping by the size of the range block), using the normal 8 transformations (four rotations and four reflections). For a fair comparison, we have used approximately the same number of codebook blocks for both the methods, which will lead to the same compression ratio. (The number of codebook blocks for Jacquin's method is $3969 \times 8 = 31752$). The results of the three methods, viz., FFT-based block matching (normal scan), FFT-based block matching (Hilbert scan), and Jacquin's method, with exhaustive search of the domain blocks, are presented in Table 1. It is seen that the SNRs obtained for FFT-based comparison methods are slightly lower than those of the conventional method, indicating that not all the circular shift transformations are 'useful' for representing image blocks. Also, among the two methods employing FFT-based block matching, the method using Hilbert scan ordering of pixels yields marginally better results. However, while the conventional method requires $31\,752 \times 64$ multiplications per range block, the number of multiplications required by the FFT-based method is about 25 times less, as we will see in Section 6.

Table 1
Comparison of PSNR of decoded images for Jacquin's method (JM), FFT-based block matching – normal scan (FFT-NS) and Hilbert scan (FFT-HS)

Image	JM	FFT-NS	FFT-HS
Lena	30.36	28.58	29.12
Baboon	24.87	24.65	25.13
Boats	30.05	28.25	29.13
Peppers	31.85	28.56	29.45

5. Faster suboptimal alternative

The block matching process can be further speeded up if we first check for a match in the magnitude of the DFT of the range and domain blocks. Since a set of $2N$ codebook vectors is obtained by circular shifting and/or reversing a domain vector, all the codebook vectors in this set have the same DFT magnitude. We can therefore make an initial comparison to choose only k domain blocks whose DFT magnitude vectors (or DFT magnitude-squared vectors) are closest to that of the range block. Thus, in the case of 8×8 range blocks, we compute the inner product of the 32 element DFT magnitude-squared vector of each range block (due to the symmetry property of the DFT of a real vector, only half the magnitude coefficients are independent) with the DFT magnitude-squared vectors of each of the 256 normalised domain blocks. Though a good match in the DFT magnitude does not imply a good overall match, simulations show that even if $k = 5$ (five best matches retained from the 256 domain blocks), there is not much difference in the quality of the match obtained in comparison with that of the exhaustive search method which requires correlation with all the domain blocks. It is shown in Section 6 that, for $k = 5$, the computational complexity is reduced by about 210 times compared to Jacquin's method. The PSNR results for this method for different values of k are tabulated for both types of scan (normal scan and Hilbert scan) in Table 2. It can be seen that for the case of Hilbert scan, even choosing only the best matching domain block ($k = 1$) does not drastically affect the PSNR of the image. This is probably due to the fact that

Table 2
PSNR of reconstructed images for various choices of k, the number of best matching domain blocks retained

	Lena		Peppers	
k	NS	HS	NS	HS
1	26.12	28.23	26.05	28.21
2	26.55	28.34	26.45	28.52
3	27.11	28.56	26.65	28.72
5	27.32	28.92	27.01	29.21
10	28.04	29.01	27.52	29.35
20	28.46	29.09	28.10	29.43
256	28.58	29.12	28.56	29.45

due to the nature of ordering of pixels in the Hilbert scan, the phase dependence is reduced (ie, the difference between vectors of adjacent shifts is not as high as in the case of the normal scan). Thus with the choice of the Hilbert scan for ordering of pixels, Fourier domain block matching, followed by FFT based correlation of each range block with a small matched set of domain blocks, yields decoded images of reasonably good quality with a drastic reduction in encoding complexity.

6. Computational complexity

We shall now compare the computational complexity of Jacquin's method with that of the two methods described in this paper. Consider an image of size $p \times p$, partitioned into $R = (p/r)^2$ range blocks of size $r \times r$, each range block containing $N = r^2$ pixels. In Jacquin's method, if the domain pool consists of blocks of size $2r \times 2r$, overlapping by the size of the range blocks, the total number of domain blocks is $[(p/r) - 1]^2 \simeq R$. Considering 8 isometric transformations of each decimated domain block, the total number of codebook blocks is $8R$. The efficient implementation of Jacquin's method by computing the inner product of each range vector with each codebook vector involves normalisation of R decimated domain vectors containing N elements each, and computation of $8R^2$ inner products of N-element real vectors. Each vector normalisation involves $2N$ multiplications, and each inner product involves N multiplications. Hence the total number of multiplications in Jacquin's method is $2RN + 8R^2N$.

In the FFT-based block-matching method of Sections 3 and 4, R/n^2 non-overlapping domain blocks of size $nr \times nr$ are decimated to the size of the range block to obtain R/n^2 domain vectors of size N, and $2NR/n^2$ codebook vectors are then generated by circular shift and inversion. In order to achieve the same compression ratio as Jacquin's method, we should use the same number of codebook vectors, viz., $8R$. Hence we choose $N/n^2 = 4$ or $n = \sqrt{N}/2$. Hence, the number of domain vectors is $4R/N$. The coding algorithm involves the following operations:

1. Normalisation of $4R/N$ domain vectors, containing N elements each.
2. Computation of the DFT of R range vectors and $4R/N$ domain vectors, each containing N elements. Among the various FFT algorithms, the split radix FFT algorithm [2] is known to have the least computational complexity. For a real vector of N elements, the number of multiplications required by this algorithm is given by

$$\mu(N) = \frac{1}{2}N(\log_2 N - 3) + 2. \tag{9}$$

3. Computation of the Hadamard product of the DFT of each range vector with the DFT (and also with the complex conjugate of the DFT) of each domain vector. The DFT of a real N-vector contains $N/2$ independent complex elements. Hence each Hadamard product involves $N/2$ complex multiplications which are equivalent to $2N$ real multiplications.
4. Computation of the inverse DFT of each Hadamard product. The total number of multiplications involved in all these operations is $8R + R(1 + 4/N)\mu(N) + 8R^2 + (8R^2/N)\mu(N)$. Hence, in comparition to Jacquin's method, the speed-up factor (SUF) of the FFT-based method is given by

$$\mathrm{SUF}_{\mathrm{FFT}} = \frac{2(4R + 1)N}{8(R + 1) + [(8R + 4)/N + 1]\mu(N)} \simeq \frac{N^2}{N + \mu(N)}. \tag{10}$$

The last approximation follows from the fact that $R \gg 1$ and $R \gg N/8$.

In the case of the sub-optimal FFT-based method of Section 5, computation of the DFT of the range vectors and the normalised domain vectors is followed by the computation of the magnitude square of the elements of all the $R(1 + 4/N)$ DFT vectors. The magnitude square computation of each DFT vector, containing $N/2$ independent complex elements, requires N multiplications. The inner product of the magnitude-squared DFT vectors of each range vector-domain vector pair involves $N/2$ multiplications. Finally, for each range vector, k domain vectors and their inverted versions are selected for performing the operations of Hadamard product in the Fourier domain followed by IDFT. Thus, the total number of multiplications involved in this method is $8R + R(1 + 4/N)\mu(N) + (1 + 4/N)RN + 2R^2 + 2kRN + 2kR\mu(N)$. Hence, in comparison to Jacquin's method, the speed-up factor of the FFT-based suboptimal (FFTSO) method is

$$\mathrm{SUF}_{\mathrm{FFTSO}} \simeq \frac{8RN}{2R + (1 + 2k + 4/N)[N + \mu(N)]}. \quad (11)$$

For an image of size 512 × 512, the SUF of the FFT method for different N and of the FFTSO method for different N and k are listed in Table 3. It is seen from Tables 2 and 3 that computational speed can be increased by a factor of 210 if we are willing to tolerate a reduction in PSNR from 30.36 dB to 28.92 dB for the 'Lena' image and from 31.85 dB to 29.21 dB for the 'Peppers' image. In general, a SUF of about 200 can be achieved at the cost of PSNR reduction by about 1.5–2.5 dB.

The values of SUF obtained by Saupe and Hartenstein [6] are also tabulated in Table 3. It is seen that these SUF values are much smaller than those achieved by the FFT and FFTSO methods. Saupe and Hartenstein have not provided PSNR values of the reconstructed images for their method for comparison. But it may be noted that the large range-block sizes (16 × 16 or larger) required to achieve a significant speed-up by the Saupe–Hartenstein method imply a corresponding degradation in the quality of the reconstructed image.

Table 3
Speed-up factors (SUF) of the FFT, FFTSO and Saupe–Hartenstein (SH) methods in comparison to Jacquin's method

	$N = 16$	$N = 64$	$N = 256$
FFT	10	25	71
FFTSO $k = 1$	64	241	441
FFTSO $k = 5$	63	210	176
FFTSO $k = 20$	62	141	54
SH	0.7	1.6	6.3

7. Conclusion

We have introduced a new technique of block matching using the FFT, which, at a little loss of quality of the reconstructed image, significantly increases the speed of the encoding process of fractal image compression. For a given compression ratio, a speed-up factor in excess of 200 can be achieved using the new technique, at the cost of a reduction in PSNR by about 1.5–2.5 dB.

References

[1] A.R. Butz, Alternative algorithm for Hilbert's space filling curve, IEEE Trans. on Computers 20 (April 1971) 924–926.

[2] P. Duhamel, Algorithms meeting the lower bounds on the multiplicative complexity of length – 2^n DFT's and their connection with practical applications, IEEE Trans. Acoust. Speech Signal Process. 38 (September 1990) 1504–1511.

[3] Y. Fisher, Introduction, in: Y. Fisher (Ed.), Fractal Image Compression: Theory and Application, Springer, New York, 1995, pp. 20–21.

[4] A.E. Jacquin, Image coding based on a fractal theory of iterated contractive image transformations, IEEE Trans. on Image Processing 1 (January 1992) 18–30.

[5] D. Saupe, R. Hamzaoui, Complexity reduction methods for fractal image compression, in: T.M. Blacklegde (Ed.), Proc. IMA Conf. in Image Processing: Mathematical Methods and Applications, Oxford University Press, 1995.

[6] D. Saupe, H. Hartenstein, Lossless acceleration of fractal image compression by fast convolution, Proc. IEEE international Conference on Image Processing, Lausanne, September 1996, pp. 185–188.