Proceedings of the 7th International Conference on Applied Informatics
Eger, Hungary, January 28–31, 2007. Vol. 1. pp. 121–128.

On speeding up fractal image compression

Tamás Kovács

Department of Informatics, Kecskemét College – Faculty of Technology,
Kecskemét, Hungary
e-mail: kovacs.tamas@gamf.kefo.hu

Abstract

In this work an improved classification algorithm is introduced, with the help of which the usually time consuming searching part of fractal image encoding speeds up considerably without loss of the reconstruction fidelity. Two parameters are used to sort image blocks into disjoint classes: the direction of the approximate first derivative and a normalized root mean square error of the fitting plane in the given block. With the help of these parameters the number of domain blocks examined for a range block is reduced dramatically. Using this algorithm the Lena 256×256 test image was encoded and the encoding results were compared to recently developed fast algorithms. The proposed method proved to be promising regarding the encoding time and reconstruction quality.

Keywords: fractal image compression

1. Introduction

The basic idea of fractal image compression (FIC) technique was introduced by Barnsley et al. [1, 2] two decades ago and the various applications and developments of this method became an area of high interest. The importance of FIC is due to that it is a promising ingredient for more effective hybrid compression methods [20, 21, 22]. By the means of the Iterated Function System proposed by Barnsley, there is a contractive transformation for each image that, according to the fix-point theorem, has the fix-point identical to the image itself. In other words, applying that transform (function) iteratively on an arbitrary starting image, the result will converge to the original image, thus the image is encoded by the transformation. Unfortunately the form of this transformation is far from trivial. A relatively simple method to establish such a transform was invented by Jacquin [3, 4], who partitioned the range and the domain image into square blocks and searched a simple affin transformation between range and domain image blocks one by one as introduced in the next section. This scheme proved to be useful however, its search process is extremely time consuming. Since then, most of the research works dealt

with the remedy of this serious drawback. The restriction of the domain pool was the main point of numerous later efforts [5, 6, 7, 8], which will be treated in more detail in the next section. Other researchers focused on improvements of the search process to make it faster by tree structure search methods [9, 10], parallel search methods [11, 12] or quad-tree partitioning of range blocks [13, 14].

The organization of this paper is as follows. In Section 2 the basic fractal image encoding method is described together with some later developments, which will be important in this work. Section 3 introduces the proposed classification algorithm, and in the last section the results of this method will be analyzed and compared to that of other fast fractal image encoding schemes.

2. Some fast variations of the basic fractal encoding algorithm

In this chapter various methods usually applied to speed up the basic algorithm are described. The contractive transformation mentioned in the introduction is constructed locally, that is, the range and the domain picture are divided into disjoint square blocks ("range blocks" and "domain blocks", respectively) and the parameters of the transformation are established for each range block. The domain block that can be mapped into a certain range block with the least error is usually called the "best matching domain block pair" for the range block. The basic point of FIC is to find this best matching pair for every range block as fast as it is possible. The basic algorithm is detailed in [3, 4, 8].

The quad-tree method proposed by Fisher [14] uses the benefits of bigger blocks without much loss of the reconstruction fidelity. In this scheme, more different block sizes are used in more consecutive steps. The local search methods are based on the assumption that the best matching domain block is situated spatially near the range block at hand, therefore the search for the proper domain block can be restricted in the vicinity of the range block [5, 6]. This, of course, results in a considerable speed up of encoding and some damage of the reconstruction fidelity as a consequence.

The so called "No search" algorithms [7, 8] do not search the domain pool but a specific domain block is appointed to be the "best matching" one. These methods produce poorer reconstruction fidelity than other partial search schemes, however, they are extremely fast, and give rise to the possibility of real time applications of fractal image encoding.

The exhausting search process can be reduced also by dividing the sets of domain and range blocks into subsets with the help of some quantitative property or parameter. By using this classification, one assumes that the parameter values of a given range block and its best matching pair are equal or close enough to each other, so the block and its pair are supposed to be in the same class (subset). The classification is a part of the pre-process that runs before the search, and in the search process the domain pool is reduced to the actual class the range block

belongs to. Wang et al. [17] and Duh et al. [18] used the edge properties of the blocks to group them into three or four classes and this resulted in a speed up ratio of 3–4. Fisher [14] divided the domain pool into 72 classes according to certain combinations of the four quadrants of the block in question. His work proved the efficiency of the classification schemes: the searching time is reduced to a few seconds without a great loss of reconstruction fidelity. Tong et al. [16] and later Wu et al. [19] used the so called standard deviation (STD) parameter to classify blocks. Wu enhanced Tong's method and achieved a still greater speed up of the encoding with compression ratio close to 1 bit per pixel.

3. The proposed method

The main purpose of the present work is to find a fast classification based encoding algorithm to reduce the encoding time to the No Search scheme's order of magnitude, while preserving a good reconstruction quality. To do this the choice of suitable block parameters is essential, which the classification is based on, and no great effort is made to optimize the developed code by other speed-up techniques. Similarly to Fisher's method [14], two different quantities are used to characterize image blocks. One of these parameters refers to the basic symmetry of image blocks, and it will be quoted here as the Approximated First Derivative (AFD) parameter. Consider the image (with size of 256×256) as a function of the form

$$f(x_i, y_i)\colon ([0,255],[0,255]) \to [0,255] \tag{3.1}$$

Though it is a discrete function, it is possible to calculate the first derivative (gradient vector) $\nabla f = (\partial_x f, \partial_y f)$ at a certain point, numerically. To do this, however, we need a numeric formula by means of which the whole image block can be characterized by the gradient vector independently from the block's size. A very simple and yet proper solution is to approximate the function values $\{f(x_i, y_i)|\ i = 1 \dots B^2\}$ of the block at hand by a linear function of the form

$$z(x, y) = q_x x + q_y y + c, \tag{3.2}$$

and use the first derivative of $z(x, y)$ as the numeric first derivative of $f(x_i, y_i)$, that is $\nabla f = (q_x, q_y)$. Regarding the symmetry of the block, only the direction of the gradient vector is of interest. Therefore its polar angle

$$\varphi = arctan\frac{q_y}{q_x}\ ;\ \varphi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \tag{3.3}$$

is used to sort the blocks into classes. In a simple manner, it can be said that the image block at hand roughly shows axial symmetry with respect to the gradient vector, and the basic point of the classification is that a given range block and its best matching domain block pair have the same axial symmetry. The AFD based sorting of blocks alone is not enough to perform a real fast algorithm. That is

why a second parameter is also needed. This is given as the normalized root mean square error (<RMSE>) of the fitting of the linear function (3.2). That is

$$< RMSE >= \frac{\sqrt{\sum_{i=1}^{B^2} [p_i - (q_x x_i + q_y y_i + c)]}}{q}, \tag{3.4}$$

where p_i is the i-th pixel value in the block and $q = \sqrt{q_x^2 + q_y^2}$ is the length of the approximated gradient vector. (The brackets $< \cdot >$, here denote the normalization by q.) Similarly to the STD parameter, which measures the deviation from the block's mean, the <RMSE> characterizes the deviation of the pixels from the fitted plane given by (3.2). After the φ and <RMSE> parameters are determined for the block at hand, they are quantized to $\{\varphi_k |\ k = 1, \ldots, K\}$ and $\{< RMSE >_l \mid l = 1, \ldots, L\}$. By means of the classification method, the set of range blocks and the domain pool is divided into separate lists identified by a pair of parameter values $(\varphi_k, < RMSE >_l)$, where $k = 1, \ldots, K$ and $l = 1, \ldots, L$, so that altogether there are $K \cdot L$ lists for the range blocks and also $K \cdot L$ lists for the domain blocks. If a given range block in question turns out to be in the list $(\varphi_k, < RMSE >_l)$, only the domain blocks contained in the list characterized by the same parameters should be compared to the range block. However, in many cases the searched for best matching domain block is in one of the eight neighboring lists $\{(\varphi_{k+i}, < RMSE >_{l+j}) |\ i = -1, 0, 1;\ j = -1, 0, 1\}$ or even in a further neighboring list of $(\varphi_k, < RMSE >_l)$. For the sake of simplicity the control parameter N_{ex} is introduced and defined as the number of the domain blocks to be examined for a range block. First the domain block list $(\varphi_k, < RMSE >_l)$ is searched, and then its neighbors, and then further neighbors, until the number of examined domain blocks exceeds N_{ex}. When applying this procedure, the choice of K and L has little effect since the speed of the algorithm and the reconstructed image quality depend on N_{ex} in the first place. Besides, Fisher's quad-tree scheme [14] is used in the present work with block sizes $B_n = \{16, 8, 4, 2\}$, and with adaptive tolerance proposed by Furao and Hasegawa [8]. The spirit of the adaptive method is also applied for N_{ex} as follows:

$$(N_{ex})_{n+1} = 2(N_{ex})_n \tag{3.5}$$

since using the greater N_{ex} value for the much more populated domain pool of smaller blocks is reasonable. The coefficient 2 in relation (3.5) was established experimentally. Furthermore, the De-Redundancy Method (DRM) and the method of treating "nearly constant blocks" introduced by Wu et al. [19] are also adopted here in an unchanged form.

4. Experiments and results

The proposed AFD-<RMSE> classification scheme, Wu's STD classification scheme [19] and Furao's No search algorithm [8] were implemented in .NET developing environment and the codes were run on a Windows XP, Pentium IV, 3 GHz

Figure 1: Original (a) and decoded (b) test images of Lena 256×256 using the proposed classification encoding method at compression ratio 1.4 bpp and with reconstruction fidelity PSNR=31.60

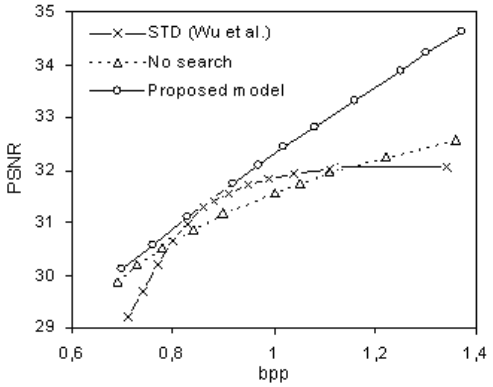

Figure 2: The PSNR versus the compression ratio (given in bit per pixel) of the proposed model and the two reference models measured on the Lena image

platform. The values of the control parameter in the AFD-<RMSE> model were the following:

$$(N_{ex})_{n+1} = \{6, 12, 24, 48\}\,. \tag{4.1}$$

The desired compression ratio was obtained by the proper choice of the tolerance threshold defined in connection with the quad-tree method. The centers of the overlapping domain blocks are situated on a square lattice, the lattice constant of which is $B/2$ pixels. Thus the total domain pool consists of $4(256/B)^2$ blocks. The reconstruction fidelity was measured by the peak signal to noise ratio (PSNR),

defined as

$$PSNR = 10 \cdot log_{10} \frac{255^2 N_p}{\sum_{i=1}^{N_p} (p_{0,i} - p_{1,i})}, \tag{4.2}$$

where $p_{0,i}$ and $p_{1,i}$ are the pixels in the original and in the reconstructed image at the same position respectively. N_p is the total number of pixels in the image. The compression ratio is expressed in bit per pixel (bpp). In the proposed scheme, encoding a $B \times B$ size image block in a 256×256 size picture requires altogether $8 + 2 \cdot log_2 \frac{256}{B} + 3 + 1 + 1$ bits (see [19]).

The implemented algorithms were compared with the help of the frequently used grayscale test image of Lena of size 256×256 (see Figure 1).

Figure 2 gives the comparison of the PSNR and compression results measured on the Lena picture. It can be seen clearly that in the higher bpp region the proposed scheme gives considerably better image quality, while for bpp values between 0.8 and 1 the three model produces PSNR results close to each other. At low bpp values the STD classification gives poorer reconstruction quality than the other two. The encoding times required to produce various PSNR levels are shown by Figure 3 The behavior of the proposed and the No Search model are similar in the sense that with increasing the PSNR level, the encoding time remains almost constant or does not grow considerably in contrast to the STD classification model (the encoding time of which grows rapidly with increasing reconstruction fidelity).

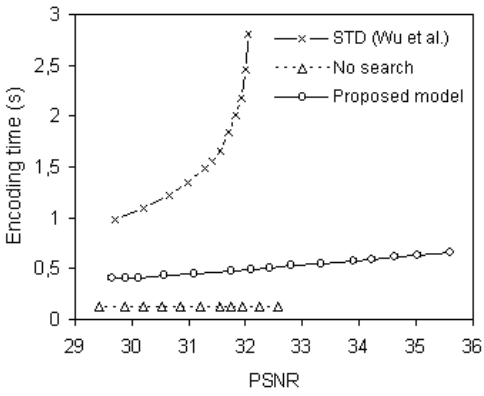

Figure 3: The encoding time versus the PSNR of the proposed model and the two reference models measured on the Lena image

The merit of the present classification algorithm is demonstrated by the fact that compared to earlier classification schemes a considerable speed up of the encoding process was achieved without loss of reconstruction fidelity, furthermore, at a higher bpp region much better image quality can be produced than that of the other

fast models. The speed increase obtained by the proposed model is the result of the number of the domain blocks examined for a range block being reduced dramatically (by the means of the parameter values given in (4.1) – this number was 90). Surely, it is possible to speed up the presented scheme by an optimized code written in a non-visual developing environment but at the same time there are further possibilities in the development of the classification method yet to be exploited as well.

References

[1] Barnsley, M. F., Fractals everywhere, *New York: Academic*, (1988).

[2] Barnsley, M. F., Sloan, A. D., A better way to compress images, *BYTE magazine*, (1988), 215–233.

[3] Jacquin, A. E., Image coding based on fractal theory of iterated contractive image transform, *IEEE Trans Image Process*, Vol. 1, (1992), 18–30.

[4] Jacquin, A. E., Fractal image coding: a review, *in: Proceedings of the IEEE*, Vol. 10, (1993), 1451–1465.

[5] Truong, T. K., Kung, C. M., Jeng, J. H., Hsieh, M. L., Fast fractal image compression using spatial correlation, *Chaos, Solitons & Fractals* , Vol. 3, (2004), 1071–1076.

[6] Monro, D. M., Dudbridge, F., Approximation of image blocks, *in: Proc. Int. Conf. Acoustics, Speech, Signal processing*, (1992), 4585–4588.

[7] Dudbridge, F., Least squares block coding by fractal functions, *in: Y. Fisher, Fractal Image Compression: Theory and Application to Digita Images, Springer, New York*, (1994), 231–244.

[8] Furao, S., Hasegawa, O., A fast no search image coding method, *Signal Processing: Image Communication*, Vol. 19, (2004), 393–404.

[9] Bani-Eqbal, B., Speeding up fractal image compression, *Proc. SPIE: Still-Image Compression*, Vol. 2418, (1995), 67–74.

[10] Hurtgen, B., Stiller, C., Fast hierarchical codebook search for fractal coding still images, *SPIE Visual Commun. PACS Med. Appl.*, (1993), 397–408.

[11] Hufnagl, C., Uhl, A., Algorithms for fractal image compression on massively parallel SIMD arrays, *Real-Time Imaging*, Vol. 6, (2000), 267–281.

[12] Vidya, D., Parthasarathy, R., Bina T. C., Swaroopa, N. G., Architecture for fractal image compression, *Journal of Systems Architecture*, Vol. 46, (2000), 1275–1291.

[13] Fisher, Y., Fractal image compression, *SIGGRAPH'92 Course Notes*, Vol. 12, (1992), 7.1–7.19.

[14] Fisher, Y., image compression: theory and applications, *Springer-Verlag, Berlin*, (1995).

[15] Oien, G. E., Lepsoy, S., A class of fractal image coders with fast decoder convergence, *in: Y. Fisher (Ed.), Fractal image compression, theory and application, Springer-Verlag, New York*, (1995), 153–174.

[16] Tong, C. S., Pi, M., Fast fractal image encoding based on adaptive search, *IEEE Trans Image Process*, Vol. 10, (2001), 1269–1277.

[17] Wang, Z., Zhang, D., Yu, Y., Hybrid image coding based on partial fractal mapping, *Signal Processing: Image Communication*, Vol. 15, (2000), 767–779.

[18] Duh, D. J., Jeng, J. H., Chen, S. Y., DTC based simple classification scheme for fractal image compression, *Image and Vision Computing*, Vol. 23, (2005), 1115–1121.

[19] Wu, X., Jackson, D. J., Chen, H. C., A fast fractal image encoding method based on intelligent search of standard deviation, *Computers and Electrical Engineering*, Vol. 31, (2005), 402–421.

[20] Shi, Y., Gu, W., Zhang, L., Chen, S., Some new methods to fractal image compression, *Comm. in Nonlinear Sci. & Numerical Simulation*, Vol. 2, (1997), 80–85.

[21] Tong, C. S., Pi, M., Analysis of a hybrid fractal-predictive-coding compression scheme, *Signal Processing: Image Communication*, Vol. 18, (2003), 483–495.

[22] Kim, T., Van Dyck, R. E., Miller, D. J., Hybrid fractal zerotree wavelet image coding, *Signal Processing: Image Communication*, Vol. 17, (2002), 347–360.

Tamás Kovács
6044 Kecskemét, Izsaki 10.
Hungary