ORDERED DECODING ALGORITHM FOR FRACTAL IMAGE COMPRESSION

Raouf Hamzaoui

Institut für Informatik, Universität Freiburg, Am Flughafen 17, 79110 Freiburg, Germany
email:hamzaoui@informatik.uni-freiburg.de

ABSTRACT

In conventional fractal image compression the decoding amounts to iterating an affine map on an arbitrary initial image until convergence. It has been observed that the convergence of the decoding can be accelerated by updating each pixel as soon as its new value is available. However, no analysis was provided on the dependence of this algorithm on the order in which the pixels are decoded. In this paper a technique is proposed where the ordering is based on the frequency with which a pixel was used in the fractal code. Simulations on several images show that this approach enables a faster convergence than the natural method where the pixels are decoded according to the order in which the ranges were encoded.

1. INTRODUCTION

One of the most interesting aspects of fractal image compression is the simplicity of the decoding. A contractive transformation T leaving the original image $\mathbf{x}^* \in \mathbf{R}^N$ almost invariant is applied iteratively on any initial image $\mathbf{x}^{(0)}$ until convergence of the sequence of iterates

$$\mathbf{x}^{(k+1)} = T(\mathbf{x}^{(k)}) \tag{1}$$

to $\mathbf{x}_T$, the fixed point of T, which is an approximation of the original image [1]. When T is an affine mapping, i.e., $T(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$, where $A = (a_{u,v})$ is an $N \times N$ matrix and $\mathbf{b} \in \mathbf{R}^N$, it has been observed independently by many researchers [2, 3, 4, 5] that the convergence of the sequence of iterates can be made faster if at each stage the newly computed components $x_u^{(k+1)}$ are used whenever available instead of $x_u^{(k)}$. More precisely, if we write the conventional iterative method (1) in the equivalent form

$$x_u^{(k+1)} = \sum_{v=1}^{N} a_{u,v} x_v^{(k)} + b_u, \tag{2}$$

then by decoding in the order $u = 1, 2, \ldots, N$ the new method corresponds to the iterative method

$$x_u^{(k+1)} = \sum_{v=1}^{u-1} a_{u,v} x_v^{(k+1)} + \sum_{v=u}^{N} a_{u,v} x_v^{(k)} + b_u.$$

A discussion of the convergence of the new method and a rationale behind its faster convergence are provided in [6, 5].

Whereas with the conventional method (2) the order in which the components $x_u^{(k+1)}$ are computed has no effect on the resulting image $\mathbf{x}^{(k+1)}$, it is clear that this is not the case when these components are progressively updated. For example, if the computations are done in the order $u = N, N-1, \ldots, 1$, then we have

$$x_u^{(k+1)} = \sum_{v=1}^{u} a_{u,v} x_v^{(k)} + \sum_{v=u+1}^{N} a_{u,v} x_v^{(k+1)} + b_u.$$

Even though distinct orderings produce different sequences of iterates, it is not difficult to see that all of them converge to the same image $\mathbf{x}_T$.

2. THE ORDERING TECHNIQUE

Let us first introduce some notations. A sampled image is a function $f : \mathcal{X} = \{0, \ldots, N_h - 1\} \times \{0, \ldots, N_v - 1\} \longrightarrow \mathbf{R}$. The elements of $\mathcal{X}$ are called *pixels*. Let $\mathcal{B}$ be a nonempty subset of $\mathcal{X}$ called *image support*. The number of pixels in $\mathcal{B}$ is called the *size* of $\mathcal{B}$. Let ψ be a one-to-one mapping from $\mathcal{X}$ to $\{1, \ldots, N\}$ where $N = N_h \times N_v$. To each image support $\mathcal{B}$ of size n we associate an $n-$dimensional vector

$$\mathbf{x}_{\mathcal{B}} = (f\psi^{-1}(b_1), \ldots, f\psi^{-1}(b_n))^T,$$

where $\psi(\mathcal{B}) = \{b_1, \ldots, b_n\} \subset \{1, \ldots, N\}$ and $b_1 < b_2 < \cdots < b_n$.

In the encoding the image support $\mathcal{X}$ is partitioned into n_R disjoint image supports $\mathcal{R}_i$, $i = 1, \ldots, n_R$, of size n_i called *ranges*. Each range vector $\mathbf{x}_{\mathcal{R}_i}$ is approximated by the linear combination

$$\widehat{\mathbf{x}_{\mathcal{R}_i}} = s_i S_i P_i \mathbf{x}_{\mathcal{D}_i} + o_i \mathbf{1}_{n_i}.$$

where

- $\mathbf{1}_{n_i} = (1, \ldots, 1)^T \in \mathbf{R}^{n_i}$.
- $\psi(\mathcal{R}_i) = \{(r_i)_1, \ldots, (r_i)_{n_i}\}$.
- $\mathcal{D}_i$ is a subset of $\mathcal{X}$ of size $m_i n_i$ called *domain* with $\psi(\mathcal{D}_i) = \{(d_i)_1, \ldots, (d_i)_{m_i n_i}\}$.
- s_i and o_i are scalar coefficients called *scaling factor* and *offset*, respectively.
- P_i is a permutation matrix of order $m_i n_i$.
- S_i is an $n_i \times m_i n_i$ *downsampling matrix* defined by

$$S_i = \frac{1}{m_i} \begin{pmatrix} \mathbf{1} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{1} \end{pmatrix},$$

where $\mathbf{1} = (1 \ldots 1)$ and $\mathbf{0} = (0 \ldots 0)$ are $1 \times m_i$ submatrices. Each approximation of a range vector $\mathbf{x}_{\mathcal{R}_i}$ defines a mapping

$$T_i : \mathbf{R}^N \to \mathbf{R}^N$$

where

$$T_i(\mathbf{x}) = s_i G_i S_i P_i F_i \mathbf{x} + o_i G_i \mathbf{1}_N.$$

Here the $m_i n_i \times N$ matrix $F_i = (f^i_{u,v})$ is defined by

$$f^i_{u,v} = \begin{cases} 1, & \text{if } (u,v) = (k, (d_i)_k), \quad k = 1, \ldots, m_i n_i; \\ 0, & \text{otherwise,} \end{cases}$$

and the $N \times n_i$ matrix $G_i = (g^i_{u,v})$ is defined by

$$g^i_{u,v} = \begin{cases} 1, & \text{if } (u,v) = ((r_i)_k, k), \quad k = 1, \ldots, n_i; \\ 0, & \text{otherwise.} \end{cases}$$

The affine operator T is given by

$$\begin{aligned} T &= \sum_{i=1}^{n_R} T_i \\ &= A\mathbf{x} + \mathbf{b}, \end{aligned}$$

where $A = \sum_{i=1}^{n_R} s_i G_i S_i P_i F_i$ is a real $N \times N$ matrix and $\mathbf{b} = \sum_{i=1}^{n_R} o_i G_i \mathbf{1}_N$ is a real column vector of dimension N.

The operator T is completely defined by specifying the range partitioning, and for each range the following four parameters: the matching domain, the permutation matrix, the scaling factor and the offset. The list of these parameters is called a *fractal code*.

In our initial implementation of the new decoding technique [4, 5] we used the natural ordering in the decoding, that is, we started by decoding and progressively updating the pixels in $\mathcal{R}_1$, then in $\mathcal{R}_2$ until reaching those in $\mathcal{R}_{n_R}$. We propose in this paper to decode and update in the order of the range frequencies in the fractal code. The notion of image support frequency is defined below and illustrated in Figure 1.

$\mathcal{R}_1$ 2	$\mathcal{R}_2$ 2	$\mathcal{R}_3$ 6	$\mathcal{R}_4$ 6
$\mathcal{R}_5$ 2	$\mathcal{R}_6$ 2	$\mathcal{R}_7$ 6	$\mathcal{R}_8$ 6
$\mathcal{R}_9$ 0	$\mathcal{R}_{10}$ 8	$\mathcal{R}_{11}$ 8	$\mathcal{R}_{12}$ 0
$\mathcal{R}_{13}$ 0	$\mathcal{R}_{14}$ 8	$\mathcal{R}_{15}$ 8	$\mathcal{R}_{16}$ 0

Figure 1: Image partitioning in 16 ranges and frequency of range occurence in the fractal code. With the proposed technique the ranges are decoded in the following order: $\mathcal{R}_{10}, \mathcal{R}_{11}, \mathcal{R}_{14}, \mathcal{R}_{15}, \mathcal{R}_3, \mathcal{R}_4, \mathcal{R}_7, \mathcal{R}_8, \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_5, \mathcal{R}_6$. The remaining ranges are decoded only at the last iteration.

Definition 1 *Let $\mathcal{D}_1, \ldots, \mathcal{D}_{n_R}$ be the domains selected in the fractal code. The frequency of an image support $\mathcal{B}$ is*

$$\frac{\sum_{i=1}^{n_R} card(\mathcal{B} \bigcap \mathcal{D}_i)}{card(\mathcal{B})},$$

where $card(\mathcal{B})$ is the number of pixels in $\mathcal{B}$.

Another approach would be to compute the frequency of each pixel and to decode pixelwise. In this case some extra memory will be used by the decoder. Note that the additional preprocessing step needed for the sorting of the ranges (or the pixels) according to their frequency is negligible.

The expected gain is twofold. First, this strategy will increase the number of pixels that will be used in their updated form yielding, hopefully, a faster convergence. Second, as a consequence of the sorting of the frequencies, the decoder can identify those ranges (or pixels) that are not present in the fractal code. These parts, as pointed out in [7, 8] in a different context, need only be decoded at the last iteration. The following proposition gives the reason why this is possible.

Proposition 1 *Let $A = (a_{u,v})$ be the $N \times N$ matrix and $\mathbf{b} = (b_u)$ the vector of dimension N associated to the affine mapping T providing the code of an image f. Suppose that the spectral radius $\rho(A) < 1$. Let $\mathbf{x}_T$ denote the fixed point of T. Now suppose that A has zero entries at the p columns $j_1, \ldots, j_p$. Let $\{i_1, \ldots, i_{N-p}\} = \{1, \ldots, N\} \setminus \{j_1, \ldots, j_p\}$. Call A' the $(N-p) \times (N-p)$ matrix obtained from A by discarding both*

columns and rows $j_1, \ldots, j_p$. Call T' the operator defined by $T'(\mathbf{x}) = A'\mathbf{x} + \mathbf{b}'$, where $b'_u = b_{i_u}$, $u = 1, \ldots, N-p$. Then we have the following:

1. $\rho(A) = \rho(A')$.
2. *Let $\mathbf{x}_{T'} = (x'_1, \ldots, x'_{N-p})^T$ denote the fixed point of T'. Then $\mathbf{x}_T = (x_1, \ldots, x_N)^T$ is given by*

$$x_{i_u} = x'_u \text{ for } u = 1, \ldots, N-p$$

and for $k = 1, \ldots, p$

$$x_{j_k} = \sum_{v \in \{1,\ldots,N-p\}} a_{j_k, i_v} x'_v + b_{j_k}.$$

Proof. 1. Without loss of generality we can assume that the p first columns are null. Thus, A can be written in the form

$$A = \begin{pmatrix} 0 & \ldots & 0 & a_{1,p+1} & \ldots & a_{1,N} \\ 0 & \ldots & 0 & a_{2,p+1} & \ldots & a_{2,N} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & a_{N,p+1} & \ldots & a_{N,N} \end{pmatrix},$$

which has the block matrix form

$$A = \begin{pmatrix} O_1 & A_1 \\ O_2 & A' \end{pmatrix},$$

where O_1 and O_2 are zero matrices of size $p \times p$ and $(N-p) \times p$, respectively. Thus, $\det(\lambda I - A) = \lambda^p \det(\lambda I - A')$, which shows that A and A' have the same spectral radius. 2. One simply verifies that the vector $\mathbf{x}_T$ is solution of the equation $A\mathbf{x}_T + b = \mathbf{x}_T$. □

By noting that pixels with zero frequency correspond to zero columns in the matrix A, the proposition implies that if the decoding procedure is convergent, then one can compute its fixed point by decoding first the pixels with nonzero frequency until convergence and then use the result to compute the value of the pixels with zero frequency in the final step. Finally observe that the result of the proposition applies also to our decoding method since a pixel with zero frequency in the fractal code corresponds also to a zero column in the iteration matrix of the new method.

3. EXPERIMENTAL RESULTS

We compare in this section the convergence of the decoding for the following three schemes.

- Scheme 1: The conventional decoding without pixel updating.
- Scheme 2: The new decoding technique where the ordering is as in the encoding.
- Scheme 3: The new decoding technique with a frequency based ordering.

First, a uniform partitioning in 8×8 range blocks is used. The domain pool consists of blocks of size 16×16 whose upper-left corners are positioned on a lattice with a vertical and horizontal spacing of $d = 8$ pixels. Thus, each domain block is a union of four range blocks. Table 1 shows the results for the 512×512 Lenna image. Here Scheme 3 is based on the range frequencies in the fractal code. Figure 2 provides statistics on the frequency of each range in the fractal code. Table 2 presents results for the 512×512 Baboon image with Scheme 3 now based on pixel frequencies (see Figure 3). Here the domain pool spacing d is equal to two pixels. Thus, domains are not unions of ranges. To give an accurate estimation of the convergence speed all computations in the previous simulations are done in double precision. However, in some implementations the values taken by the pixels at each iteration are rounded to integers to save memory space. This does not hinder the success of our methods which show the same improvements when rounding is used. Figure 4

Iteration	Scheme 1	Scheme 2	Scheme 3
1	73.1453	59.6675	46.8337
2	42.2518	25.1607	17.1060
3	24.6055	9.5244	4.7620
4	14.0406	2.3372	0.9230
5	4.5585	0.3805	0.1472
6	1.3235	0.0740	0.0287
7	0.3726	0.0168	0.0070
8	0.1080	0.0037	0.0020

Table 1: Convergence of the decoding for the three methods for the 512×512 Lenna image. The values given in the table are the root mean square errors between the fixed point and successive iterates of an initial black image $\mathbf{x}^{(0)} = (0, \ldots, 0)^T$.

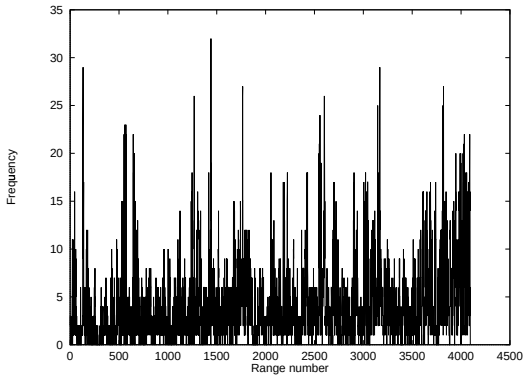

Figure 2: Range frequency in the fractal code.

Iteration	Scheme 1	Scheme 2	Scheme 3
1	96.1099	80.8097	64.4852
2	66.8111	38.9070	25.150
3	39.8743	13.3733	6.8247
4	19.7826	3.4775	1.4501
5	7.8572	0.7364	0.2827
6	3.0382	0.1668	0.0574
7	1.1755	0.0360	0.0120
8	0.4671	0.0074	0.0026

Table 2: Convergence of the decoding for the three methods for the 512×512 Baboon image. Scheme 3 is based on the pixel frequencies in the fractal code.

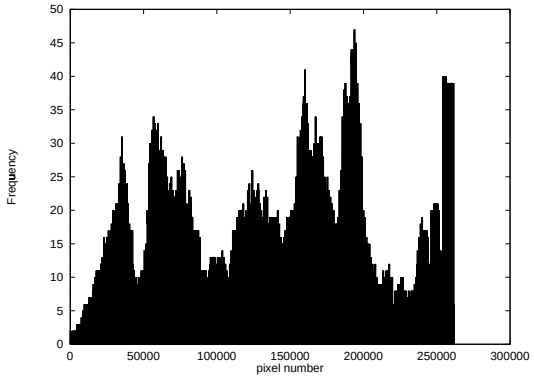

Figure 3: Pixel frequency in the fractal code. Due to the limitations of the plotting device pixels with zero frequencies do not appear on the graph.

shows the second iterate produced by the three decoding schemes for the quadtree encoded 512×512 Bridge image when rounding is used at each iteration.

Even though the decodings with the frequency based orderings do not converge much faster than Scheme 2, they still may be preferred since at each step the iterates provide a better approximation. Similar results were obtained for all other tested images and domain pools.

4. CONCLUSION

We have shown that the decoding in fractal image compression can be improved by combining a pixel update scheme with an ordering technique based on the frequency of the ranges (or the pixels) in the fractal code. Besides the computational gains inherent to the noniterative decoding of zero frequency ranges (or pixels), experimental results indicate that the new method allows better approximations at each iteration step and a faster convergence.

5. REFERENCES

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

[2] Kaouri, H., *Fractal coding of still images,* in: *IEE 6th International Conference on Digital Processing of Signals in Communications,* pp. 235–239, 1991.

[3] Nelson, M., Gailly, J.-L., *The Data Compression Book*, M & T books, New York, 1995.

[4] Hamzaoui, R., *Decoding algorithm for fractal image compression,* Electronics Letters 32,14 (1996) 1273–1274.

[5] Hamzaoui, R., *Fast decoding algorithms for fractal image compression,* in: *Proceedings of the conference Fractals in Engineering*, Arcachon, June 1997.

[6] Hamzaoui, R., *Fast decoding algorithms for fractal image compression,* Technical Report 86, Institut für Informatik, University of Freiburg, March 1997.

[7] Hürtgen, B., Simon, S. F., *On the problem of convergence in fractal coding schemes,* in: *Proc. ICIP-94 IEEE International Conference on Image Processing,* Vol. 3, Austin, 1994.

[8] Domaszewicz, J., Vaishampayan, V. A., *Graph-theoretical analysis of the fractal transform,* in: *Proc. ICASSP-1995 IEEE International Conference on Acoustics, Speech and Signal Processing,* Vol. 4, Detroit, 1995.

Figure 4: The fixed point image (upper left, PSNR = 29.85 dB), and the second iterate image with the conventional decoding (upper right, PSNR = 13.57 dB), the new method with the natural ordering (lower left, PSNR = 19.74 dB), and the new method with the ordering based on pixel frequencies (lower right, PSNR = 21.83 dB).