FAST CODE ENHANCEMENT WITH LOCAL SEARCH FOR FRACTAL IMAGE COMPRESSION

Raouf Hamzaoui, Dietmar Saupe, Michael Hiller

Universität Leipzig, Institut für Informatik, Augustusplatz 10–11, 04109 Leipzig, Germany

ABSTRACT

Optimal fractal coding consists of finding in a finite set of contractive affine mappings one whose unique fixed point is closest to the original image. Optimal fractal coding is an NP-hard combinatorial optimization problem. Conventional coding is based on a greedy suboptimal algorithm known as collage coding. In a previous study, we proposed a local search algorithm that significantly improves on collage coding. However, the algorithm, which requires the computation of many fixed points, is computationally expensive. In this paper, we provide techniques that drastically reduce the time complexity of the algorithm.

1. INTRODUCTION

The rate-distortion results of the best fractal coders are inferior to those of the state-of-the-art in image compression [14]. However, the potential of fractal image compression has not been fully exploited because current fractal schemes do not find *optimal* codes. In fractal image compression, the code is a representation of a contractive affine mapping in a complete metric space of digital images $(\mathcal{F}, d)$ [5]. The encoder finds a contractive affine mapping $T : \mathcal{F} \to \mathcal{F}$ of which the fixed point f_T approximates the original image. The decoder computes f_T as the limit point of the sequence of iterates $\{f_n\}_{n\geq 0}$ where $f_{n+1} = T(f_n)$, and f_0 is an arbitrary starting image. Given a target image f^*, a large finite set $\mathcal{T}$ of contractive affine mappings, and a number of bits r, an *optimal* encoding is a solution to the constrained minimization problem

$$\min_{T\in\mathcal{T}} \Delta(f^*, f_T) \text{ subject to } \mathrm{len}(c(T)) \leq r.$$

Here $\Delta(f^*, f_T) \geq 0$ is the reconstruction error, and $\mathrm{len}(c(T))$ is the length of the code $c(T)$ of the mapping T. Conventional coding [5], also known as collage coding, finds only a suboptimal solution. Several researchers [2, 1, 9, 4, 10, 13, 8] recognized this problem and found better though still suboptimal solutions. The most successful approach uses a local search technique [8]. The algorithm starts from a code computed by collage coding and keeps on improving this code with a heuristic introduced in [1, 10] until convergence to a locally optimal solution. For quadtree-based image partitions [6], typical PSNR improvements over collage coding range from 0.2 dB to 0.8 dB. Unfortunately, the algorithm is time expensive because many iteration steps are needed for its convergence. Moreover, the evaluation of the reconstruction error, which is needed at each iteration step, requires the computation of the fixed point of an affine mapping of a high dimensional space.

Published in the 2000 International Conference on Image Processing (ICIP-2000), Sept. 10-13, 2000, Vancouver, BC.

The main purpose of this paper is to present an efficient implementation of the algorithm. The basic idea is to exploit the dependence graph [3] of the code.

The rest of the paper is organized as follows. In Section 2, we introduce terminology, present a generic fractal scheme and discuss previous work. In Section 3, we show how the local search algorithm can improve the state-of-the-art fractal scheme of [12]. In Section 4, we present four new techniques that accelerate the local search algorithm. Section 5 contains numerical results of an implementation based on these techniques. In the last section, we discuss the results.

2. TERMINOLOGY

Let $\mathcal{F}$ be the vector space of digital images $f : I = \{0, 1, \ldots, N-1\} \times \{0, 1, \ldots, N-1\} \to \mathbb{R}$. Then $\mathcal{F}$ is a complete metric space for the L_∞ metric

$$d_\infty(f, g) = \|f - g\|_\infty = \max_{i,j} |f(i,j) - g(i,j)|.$$

Let $B \subset I$ be an $n \times n$ block. We denote by $\mathbf{x}_{f|B}$ the column vector formed by stacking the pixel intensities of B row by row, left to right, and top to bottom. Let $\{R_1, \ldots, R_{n_R}\}$ be a partition of I into pairwise disjoint $2^n \times 2^n$ square blocks called *range blocks*. Let $\mathcal{D} = \{D_1, \ldots, D_{n_D}\},\ D_i \subset I$, be a set of $2^{n+1} \times 2^{n+1}$ blocks called *domain blocks*. Let $\mathcal{S} = \{s_1, \ldots, s_{n_s}\}$ be a set of real numbers called *scaling factors*. Let $\mathcal{O} = \{o_1, \ldots, o_{n_o}\}$ be a set of real numbers called *offsets*. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_{n_P}\}$ be a set of permutation matrices of order 2^{2n}. To each parameter tuple

$$((D(1), s(1), o(1), P(1)), \ldots, (D(n_R), s(n_R), o(n_R), P(n_R)))$$

where $(D(i), s(i), o(i), P(i)) \in \mathcal{D} \times \mathcal{S} \times \mathcal{O} \times \mathcal{P}$, we associate a transformation $T : \mathcal{F} \to \mathcal{F}$ such that for $f \in \mathcal{F}$ and all $i \in \{1, \ldots, n_R\}$

$$\mathbf{x}_{T(f)|R_i} = s(i)P(i)A\mathbf{x}_{f|D(i)} + o(i)\mathbf{1}.$$

Here $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^{2^{2n}}$, and A is a $2^{2n} \times 2^{2(n+1)}$ *downsampling matrix*. The set of all such transformations T is denoted by $\mathcal{T}$. Moreover, we call $\mathcal{D}$ a *domain pool* and $D(i), s(i), o(i), P(i)$ the *fractal parameters* of range block R_i. If the scaling factors are restricted to the interval $[-s_{\max}, s_{\max}]$, $s_{\max} < 1$, then T is a contraction in the L_∞ metric and the decoding $f_0 \to T(f_0) \to T(T(f_0)) \to \cdots$ is convergent to the fixed point f_T independently of the initial image f_0 [5].

Let us assume that the codes of the transformations T have the same length. Then an optimal transformation T_{opt} is one that minimizes the reconstruction error

$$E(T) = \Delta(f^*, f_T) = \|f^* - f_T\|_2^2 = \sum_{i,j} (f^*(i,j) - f_T(i,j))^2$$

over all feasible parameter tuples. There are $(n_D n_s n_o n_P)^{n_R}$ such tuples. Thus, finding T_{opt} by enumeration is impractical for large n_R. Moreover, Ruhl and Hartenstein [11] proved that finding an optimal solution is an NP-hard problem. Usually, a suboptimal solution is found by a greedy algorithm known as collage coding, which consists of minimizing the *collage error* $\|f^* - T(f^*)\|_2^2$ instead of the reconstruction error $\|f^* - f_T\|_2^2$. Collage coding simplifies the optimization problem because it reduces to the independent minimization of the n_R local collage errors

$$\|\mathbf{x}_{f^*_{|R_i}} - (s(i)P(i)A\mathbf{x}_{f^*_{|D(i)}} + o(i)\mathbf{1})\|_2^2$$

for $(D(i), s(i), o(i), P(i)) \in \mathcal{D} \times \mathcal{S} \times \mathcal{O} \times \mathcal{P}$.

3. LOCAL SEARCH ALGORITHM

Hamzaoui *et al.* [8] proposed a local search algorithm that iteratively improves an initial code found by collage coding. The algorithm is as follows:

1. **Initialization**: Let M be a maximum number of trials. Set $n := 0$, $i := 0$, and $k := 0$. Find an initial transformation T_0 by collage coding. Compute the fixed point of T_0. Let n_R be the number of range blocks in the partition.
2. Let $r := 1 + (i \bmod n_R)$. Determine for the range block R_r new fractal parameters by minimizing the error
   $$\|\mathbf{x}_{f^*_{|R_r}} - (s(r)P(r)A\mathbf{x}_{f_{T_n|D(r)}} + o(r)\mathbf{1})\|_2^2$$
   for $(D(r), s(r), o(r), P(r)) \in \mathcal{D} \times \mathcal{S} \times \mathcal{O} \times \mathcal{P}$. Set $i := i + 1$.
3. Construct a candidate transformation T_c by changing the fractal parameters of only range block R_r according to the result of Step 2.
4. Compute the fixed point of T_c and the reconstruction error $E(T_c)$.
5. If $E(T_c) < E(T_n)$, set $T_{n+1} := T_c, n := n + 1, k := 0$. Otherwise set $k := k + 1$.
6. If $(i \leq M$ and $k < n_R)$ go to Step 2. Otherwise stop.

When M is set to ∞, the algorithm stops at a local minimum. In Step 2, the minimization problem is treated with a least squares technique as in collage coding [5].

For clarity of presentation, our generic fractal scheme was restricted to uniform partitions into square blocks. However, fractal coding provides better rate-distortion and subjective quality results when the partition is adapted to the contents of the image. This includes quadtree partitions, rectangular partitions, and more sophisticated partitions based on region growing [12]. Because the range blocks may have various sizes and shapes, several domain pools have to be used. The local search algorithm can be extended in a straightforward way to such schemes by searching in Step 2 for domain blocks in the appropriate domain pools.

The local search algorithm can improve the PSNR of the codes obtained with Fisher's quadtree scheme [6] by up to 0.8 dB [8]. We now use the local search technique to improve codes computed by the efficient fractal scheme of [12]. In this scheme, the image support is first partitioned into uniform *atomic* square range blocks. For each range block, the d sets of fractal parameters that yield the smallest local collage error are determined. Then the two adjacent range blocks whose merging yields the least increase of the total collage error are iteratively merged until a partition into n_R range blocks is obtained. When two adjacent (parent) range blocks are merged, a set of candidate domain blocks is formed by appropriately extending the size of each of the $2d$ domain blocks corresponding to the parent range blocks and retaining the d ones that yield the smallest local collage error. Table 1 gives the PSNR performance of the code in [12] and that of the local search algorithm after $i = n_R$ trials (third column) and after convergence (last column) for three test images. In Step 2 of the algorithm, for each range block, the search for a new domain block was restricted to the local domain pool consisting of the final set of d domain blocks. In our experiments, we used the following settings. The atomic block size was 8 × 8, $d = 50$, and $n_R = 1000$. Figure 1 shows the resulting partition for the 512 × 512 Barbara image. The experiments show that the local search algorithm improved the reconstruction fidelity of the fractal scheme of [12] by about 0.25 dB. Moreover, most of the gain was obtained after the first n_R trials. We had similar PSNR gains for other values of d and n_R.

Image	[12] PSNR (dB)	$i = n_R$ PSNR (dB)	Converg. PSNR (dB)
512 × 512 Barbara	24.02	24.25	24.27
512 × 512 Lenna	29.21	29.39	29.39
512 × 512 Boat	27.70	27.91	27.94

Table 1: Reconstruction fidelity with local search for the fractal scheme based on region-growing ([12]). The image partition consists of 1000 range blocks.

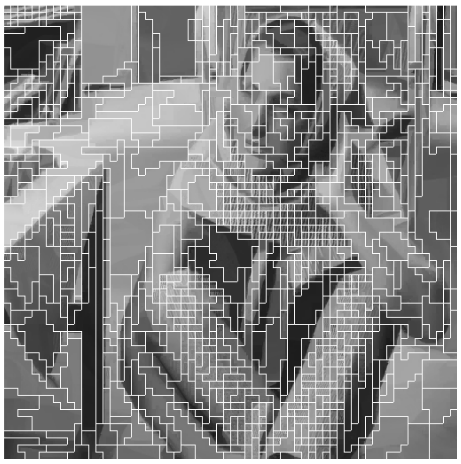


Figure 1: Partition into 1000 range blocks for an atomic block size of 8 × 8. The compression ratio is 63.12:1.

4. COMPLEXITY REDUCTION OF THE ALGORITHM

In [8], the algorithm was accelerated as follows. In Step 4, the fixed point of T_c is computed by starting the iterations from the fixed point of the current transformation T_n. This allows fast convergence because these fixed points are close to each other. Moreover, a Gauss-Seidel like iteration [7] is used, where only one image array is stored, and the new pixel intensities are used as soon as they become available. We now present four new techniques, which further reduce the time complexity of the algorithm.

4.1. No decoding

In Step 2 of the algorithm, it may happen that the new fractal parameters of range block R_r are equal to the current ones. In this case, there is no need to compute the fixed point of the candidate mapping T_c. Thus, we just set $k := k + 1$ and go to Step 6.

4.2. Ordering of the range blocks

The local search algorithm is dependent on the ordering of the range blocks. In [8], the range blocks $R_r,\ r \in \{1, \ldots, n_R\}$ were considered in the order given by the fractal code. We have found out, however, that the reconstruction error decreases faster in the initial steps if the range blocks are ordered according to decreasing error $\|\mathbf{x}_{f^*_{|R_r}} - \mathbf{x}_{f_{T_0\,|R_r}}\|_2^2$.

4.3. Decoding with dependence graph

The main idea for accelerating the local search algorithm exploits the fact that T_c and T_n differ only in the fractal parameters of range block R_r. Thus, in Step 4, if we start from the current fixed point f_{T_n} and apply the operator T_c once, then only the pixel intensities in R_r have to be updated, which saves many unnecessary computations. If we now apply T_c to the first iterate $T_c(f_{T_n})$, then only the pixel intensities of the range blocks whose domain blocks overlap R_r have to be updated. Note that these range blocks may include R_r. This procedure is repeated for the next iterations until convergence to f_{T_c}. The decoding relation between the range blocks was studied by Domaszewicz and Vaishampayan [3] who introduced the notion of *dependence graph* of the code, which is defined as follows. We say that a block $D \subset I$ *overlaps* a block $R \subset I$ if $D \cap R \neq \emptyset$. A range block R_j is called a *child* of a range block R_i if the domain block that encodes R_j overlaps R_i. The range block R_i is then called a *parent* of R_j. A *dependence graph* of a code is a directed graph $\mathcal{G} = (V, E)$ where each vertex $R_i \in V$ is a range block in the image partition, and an ordered pair of vertices (R_i, R_j) is in E if R_j is a child of R_i. Note that each range block has at least one parent, but not all range blocks have children. The decoding $f_{T_n} \to T_c(f_{T_n}) \to T_c(T_c(f_{T_n})) \to \cdots \to f_{T_c}$ can be seen as a breadth-first traversal of the dependence graph, starting from vertex R_r. The first iteration corresponds to visiting R_r and for $n \geq 2$, iteration n corresponds to visiting the children of all vertices visited at iteration $n-1$. Note that only vertices that are connected to R_r are visited. Note also that at a given iteration, a vertex needs to be visited only once.

In fact, instead of the usual operator T_c, we use the one associated to the faster Gauss-Seidel like iteration where the new pixel intensities are used as soon as they are available. In this situation, a breadth-first traversal of the dependence graph as above corresponds to a Gauss-Seidel like iteration where the range blocks are ordered according to decreasing distance to the vertex R_r. We recall that the distance between two vertices R_r and R_i in the same connected component of a graph is the length of the shortest simple path from R_r to R_i.

The dependence graph is stored as an array of linked lists. Once the fractal parameters of range block R_r are changed, the dependence graph is updated as follows.

1. Determine the current parents of vertex R_r.
2. Go through the linked lists of these vertices and remove R_r from each list.
3. Determine the new parents of vertex R_r.
4. Insert R_r in the linked list of each new parent.

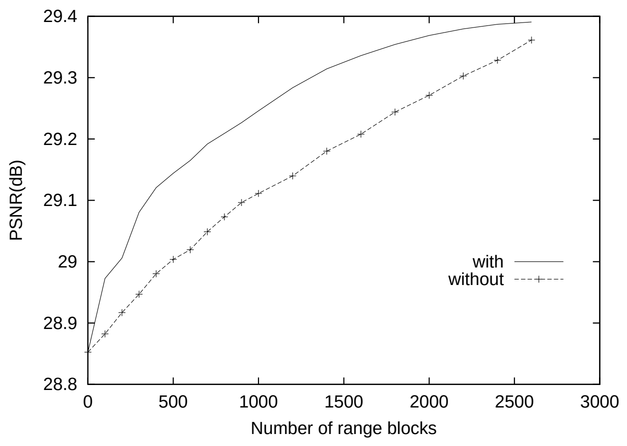
To find the parents of vertex R_r, one has to identify the range blocks that are overlapped by the domain block that encodes R_r. But if this domain block is selected another time, then the same operation has to be repeated. Thus, one can save computing time by determining in a preprocessing step for all domain blocks in the domain pool the range blocks that they overlap.

4.4. Computation savings for unchanged domains

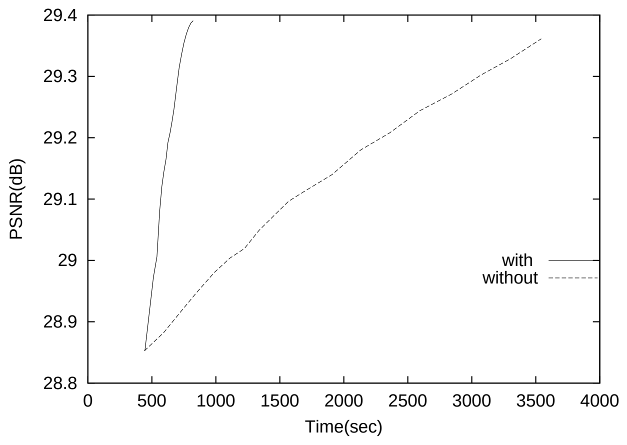
The least squares approach used in Step 2 of the algorithm requires the computation of the sum of the pixel intensities and the sum of the squared pixel intensities of f_{T_n} for all domain blocks in the domain pool [6]. But according to subsection 4.3, only a few range blocks have to be processed. Thus, for $f_{T_{n+1}}$, we compute these sums only for the domain blocks that overlap these range blocks.

5. RESULTS

This section illustrates the relevance of the proposed acceleration techniques. The test image was the 8 bpp 512 × 512 Boat image. The initial coding was done with Fisher's quadtree scheme [6]. The smallest range block size was 4 × 4, and the largest range block size was 32×32. For each range block size, the domain pool consisted of the square blocks of the image support whose linear size is twice that of the range blocks and whose upper-left pixels are situated on locations (i, j), where $i \equiv 0(\bmod 4)$ and $j \equiv 0(\bmod 4)$. Thus, there were $n_{D_1} = 12769$ domain blocks of size 64 × 64, $n_{D_2} = 14641$ domain blocks of size 32 × 32, $n_{D_3} = 15625$ domain blocks of size 16 × 16, and $n_{D_4} = 16129$ domain blocks of size 8 × 8. We slightly modified Fisher's coder in a way that only one permutation, namely the identity was used (thus $n_P = 1$). For each block size, the parameters n_s and n_o were equal to 32 and 128, respectively. The root mean square (rms) threshold was equal to 14. With these settings, the image was partitioned into $n_R = 2686$ range blocks. In the computation of the fixed point of T_c, we stopped the iteration when the rms error between two consecutive image iterates was less than the threshold 0.2. Figure 2 (a) shows the PSNR yielded by the solution T_n as a function of the number of range blocks considered (counter i in the algorithm). Figure 2 (b) shows the PSNR as a function of time for the same experiment. The CPU time was measured on an SGI O2 running an MIPS R10000 195 MHz processor and having a main memory size of 192 Mbytes. The curves labeled ("with") correspond to an implementation of the local search algorithm which includes all four acceleration techniques. The curves labeled ("without") correspond to a naive implementation which does not include these techniques (the techniques of [8] were not



(a)



(b)

Figure 2: (a) PSNR versus number of range blocks considered for the 512 × 512 Boat image. (b) PSNR versus time for the 512 × 512 Boat image. The curves labeled "with" correspond to an implementation of the algorithm that includes the acceleration techniques.

used either). Note that in Figure 2 (a) only the effect of the differing ordering of the range blocks can be seen (see 4.2). Although both curves eventually reached about the same quality, the new ordering allowed a steeper increase of the PSNR in the beginning. This is advantageous when the algorithm is stopped after a few steps. The initialization (collage coding) took 445 seconds and yielded a PSNR of 28.85 dB at a compression ratio of 29.19:1. The fast local search algorithm was able to increase the PSNR by 0.3 dB in about 150 seconds. The PSNR gain reached 0.45 dB after 100 more seconds. Table 2 illustrates the computation savings due to the techniques of 4.1 and 4.3. The first column gives the current number of range blocks considered by the algorithm (counter i). The second column gives the number of blocks with unchanged fractal parameters. The last column gives the average number of range blocks that were visited in the breadth-first traversal of the dependence graph during the computation of f_{T_c}. For example, for $i = n_R$, this average number was $\frac{7487}{2686-568} = 3.53$.

Counter i in algorithm	Number of ranges with unchanged parameters (4.1)	Average number of processed ranges (4.3)
$n_R(2686)$	568 (21 %)	3.53
$2n_R(5372)$	2261 (42 %)	4.32

Table 2: Computation savings due to the techniques of 4.1 and 4.3.

6. CONCLUSION

A fractal code obtained with a quadtree scheme [6] can be improved with a local search algorithm that successively modifies the code parameters of the range blocks and updates these parameters when the decoding error is decreased [8]. In this paper, we showed that the algorithm is also useful for a state-of-the-art fractal scheme based on more adaptive partitions [12]. Moreover, we proposed an efficient implementation of the algorithm. The basic idea is to see that when the code parameters of only one range block are modified, then only a few range blocks have to be processed by the decoder. The price for the speed-up of the algorithm is additional memory needed in particular for storing the dependence graph [3] of the fractal code.

7. REFERENCES

[1] Barthel, K. U., Voyé, T., *Adaptive fractal image coding in the frequency domain,* in: *Proc. Int. Workshop on Image Processing: Theory, Methodology, Systems and Applications,* Budapest, June 1994.

[2] Domaszewicz, J., Vaishampayan, V. A., *Iterative collage coding for fractal compression,* in: *Proc. IEEE ICIP-94,* vol. 3, Austin, Texas, Nov. 1994.

[3] Domaszewicz, J., Vaishampayan, V. A., *Graph-theoretical analysis of the fractal transform,* in: *Proc. IEEE ICASSP-1995,* vol. 4, Detroit, 1995.

[4] Dudbridge, F., Fisher, Y., *Attractor optimization in fractal image encoding,* in: *Proc. of the Conference Fractals in Engineering,* Arcachon, June 1997.

[5] Fisher, Y., *Fractal Image Compression — Theory and Application,* Springer-Verlag, New York, 1994.

[6] Fisher, Y., *Fractal image compression with quadtrees,* in: *Fractal Image Compression — Theory and Application,* Y. Fisher (ed.), Springer-Verlag, New York, 1994.

[7] Hamzaoui, R., *Fast iterative methods for fractal image compression,* Journal of Mathematical Imaging and Vision 11,2 (1999) 147–159.

[8] Hamzaoui, R., Hartenstein, H., Saupe, D., *Local iterative improvement of fractal image codes,* Image and Vision Computing 18 (2000) 565–568.

[9] Hürtgen, B., *Performance bounds for fractal coding,* in: *Proc. IEEE ICASSP-1995,* vol. 4, Detroit, 1995.

[10] Lu, N., *Fractal Imaging,* Academic Press, 1997.

[11] Ruhl, M., Hartenstein, H., *Optimal fractal coding is NP-hard,* in: *Proc. DCC'97,* J. A. Storer and M. Cohn (eds.), IEEE Comp. Soc. Press, pp. 261–270, March 1997.

[12] Ruhl, M., Hartenstein, H., Saupe, D., *Adaptive partitionings for fractal image compression,* in: *Proc. IEEE ICIP-97,* vol. 2, Santa Barbara, California, Oct. 1997.

[13] Vrscay, E. R., Saupe, D., *Can one break the "collage barrier" in fractal image coding,* in: *Fractals: Theory and Applications in Engineering,* M. Dekking, J. L. Vehel, E. Lutton and C. Tricot (eds.), pp. 307–323, Springer-Verlag, London, 1999.

[14] Wohlberg, B. E., de Jager G., *A review of the fractal image coding literature,* IEEE Transactions on Image Processing 8(12) (1999) 1716–1729.