Rate-Distortion Based Fractal Image Compression

Raouf Hamzaoui and Dietmar Saupe

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

Abstract: In fractal image compression a contractive affine mapping whose fixed point is an approximation to the original image has to be determined. Usually, this mapping is found in a heuristic way. In this paper, we discuss rate-distortion based fractal image coding where the code corresponding to the mapping is optimal in the sense that it guarantees the lowest collage error (distortion between the original image and its first iterate) over a large set of admissible codes subject to a rate constraint. We show how optimal codes can efficiently be obtained. We begin with a fine scale partition of the image which gives a fractal encoding with many bits and a low collage error. The partition is hierarchical, thus, corresponds to a tree. We use a pruning strategy based on the generalized BFOS algorithm where we extract subtrees corresponding to partitions and fractal encodings which are optimal. We give results for fractal image compression with rectangular partitions. We also provide a comparison with heuristic techniques and review other rate-distortion based approaches.

1 Introduction

In fractal image compression a contractive affine mapping W of a complete metric space of images (X, δ) has to be found such that its unique fixed point $f_W = W(f_W)$ is an approximation to the original image f [1, 16, 8]. The decoding is based on the contraction mapping principle, which guarantees that f_W can be computed as the limit point of the sequence of iterates $\{f_n\}$, where $f_{n+1} = W(f_n)$ and f_0 is an arbitrary initial image. In practice, images are digitized both spatially and in amplitude. In this situation, the metric space (X, δ) is taken as the n-dimensional space $\mathbf{R}^n$ together with a distance derived from a vector norm (e.g. the L_2 norm).

The affine mapping W is composed of blockwise affine transformations each approximating a block of the image (*range block*) by a larger block from the same image (*domain block*) [8]. W is given by a partition of the original image f into disjoint range blocks $R_1, \ldots, R_{n_R}$, and for each range block R a *transformation list* consisting of:

- a domain block D larger than the range block and taken from the same image f.

- a *scaling factor* $s \in \{s_1, \ldots, s_{n_s}\} \subset [-s_{max}, s_{max}] \subset \mathbf{R}$.
- an *offset* $o \in \{o_1, \ldots, o_{n_o}\} \subset \mathbf{R}$.

Each transformation list associates to the range block R a *local collage error*

$$E_c(R, D, s, o) = \|R - (sAD + o\mathbf{1})\|_2^2, \tag{1.1}$$

where the operator A downsamples a domain block by pixel averaging to match the range block size, and $\mathbf{1}$ is the flat block with intensity 1 at every pixel. Here blocks are considered as column vectors by reading their pixel intensities row by row, left to right and top to bottom.

According to the *Collage Theorem* [1], we have

$$\|f - f_W\| \leq \frac{1}{1-L}\|f - W(f)\|,$$

where $0 \leq L < 1$ is the contraction factor of W for a norm $\|\cdot\|$ in which W is contractive. Thus, by finding for each range block in the partition a transformation list minimizing its collage error, one hopes to generate a fixed point of W close enough to the original image. Direct optimization of the reconstruction error $\|f - f_W\|_2$ is an intractable problem as shown by Ruhl and Hartenstein in [21].

In this paper we call optimal rate-distortion fractal code a solution of the discrete constrained optimization problem

$$\min_{W \in \mathcal{W}} d(W) = \sum_{i=1}^{n_R} E_c(R_i, D, s, o) \tag{1.2}$$

subject to

$$l(c(W)) \leq r_0$$

where

- $\mathcal{W}$ is the finite set of contractions obtained by considering all admissible range partitions, and for each range block in the partition all admissible domain blocks, scaling factors and offsets,
- $c(W)$ is a variable length binary code of W,
- $l(c(W))$ is the bit length of the code $c(W)$, and
- r_0 is a given bit budget.

The binary code $c(W)$ consists of bits for the range partition, the domain block addresses, and the scaling factor and offset indices.

In principle, since $\mathcal{W}$ is a finite set, the above problem could be solved by exhaustive search. However, due to the huge number of admissible solutions, this approach is infeasible. For this reason, only a limited number of partition types are used: uniform, quadtree, rectangular, triangular, other polygonal, and irregular partitions [8].

If the image partition is fixed and if for a range block the codewords of respectively all domain blocks, all scaling factors and all offsets are of the same length, then an almost optimal fractal code can be found as follows. First, for a domain block D_k, use the method of least squares to compute the best fitting scaling and offset parameters s and o. Then, quantize theses parameters and compute the collage error

$$E_c(R, D_k) = E_c(R, D_k, \overline{s}, \overline{o})$$

with the quantized parameters $\overline{s}, \overline{o}$. Finally, select the domain block that minimizes $E_c(R, D_k)$.

If the partition is not fixed, heuristic methods, which yield suboptimal solutions have been developed. The most common approach is proceeding in a *top-down* direction, i.e., larger ranges are split into several smaller ones, which are subject to further subdivision [9, 3, 10]. A range block is subdivided if an estimate of the collage error for this block is above some threshold. On the other hand, one may start with a fine partition (e.g., a uniform one consisting of small range blocks, all of the same size), and then iteratively merge some of the ranges [15, 22]. In this case the range block merging can be organized so as to minimize the overall collage error. In contrast to these partition methods based on local collage error computations, one may design a partition solely based on some segmentation methods from image processing, thereby not searching for any range-domain pairs until the partition is complete. The advantage of this approach is increased speed [26, 6].

In this paper, we show that optimal solutions to problem (1.2) can be efficiently determined in a general context. The tool to achieve these rate-distortion optimal codes is the generalized BFOS algorithm [5], which we review in the following section. In Section 3 we apply the method to fractal image coding with rectangular (HV) partitions. Section 4 contains simulation results where greedy techniques are compared to our optimal strategy. In Section 5 we discuss extensions of the method and in Section 6 we briefly present other rate-distortion based fractal coding schemes.

2 Optimal tree pruning using the generalized BFOS algorithm

The terminology and notation used in this section closely follow those in [12] where more details on the algorithm can be found.

Let $\mathcal{T}$ be a tree with root node n_0. A *branch* $\mathcal{T}_n$ of a tree $\mathcal{T}$ is a subtree of $\mathcal{T}$ rooted at a node n such that the leaf nodes of the subtree are also leaf nodes of $\mathcal{T}$. We say that $\mathcal{S}$ is a *pruned subtree* of $\mathcal{T}$, and write $\mathcal{S} \preceq \mathcal{T}$ if $\mathcal{S}$ is a subtree of $\mathcal{T}$ with the same root node n_0. A tree functional u is a real function defined on the set of all subtrees of $\mathcal{T}$.

Let r and d denote two tree functionals (such as rate and distortion). Assume that for each $\mathcal{S} \preceq \mathcal{T}$ we have $r(\mathcal{S}) \geq r_{min}$. Then we call an *optimal pruned subtree* any solution of the discrete optimization problem

$$\min_{\mathcal{S} \preceq \mathcal{T}} d(\mathcal{S})$$

subject to

$$r(\mathcal{S}) \leq r_0,$$

where $r_0 \geq r_{min}$

When $\mathcal{T}$ has a large number of nodes, finding a solution by full search is impractical. Fortunately, if we assume that the tree functionals r and d are monotonic and linear, with r monotonically increasing and d monotonically decreasing, then the problem can be efficiently treated by the algorithm proposed by Chou *et al.* [5, 12], which is a generalization of the BFOS algorithm of Breiman, Friedman, Olshen, and Stone [4]. Here a tree functional u is said to be *monotonically increasing* (resp. *monotonically decreasing*) if

$$\mathcal{S}' \preceq \mathcal{S} \Rightarrow u(\mathcal{S}') \leq u(\mathcal{S}) \ \ (\text{resp. } u(\mathcal{S}') \geq u(\mathcal{S})),$$

and it is said to be *linear* if its value is given by the sum of its values at the leaf nodes, that is, if

$$u(\mathcal{S}) = \sum_{n \in \tilde{\mathcal{S}}} u(n),$$

where $\tilde{\mathcal{S}}$ is the set of leaf nodes of the subtree $\mathcal{S}$.

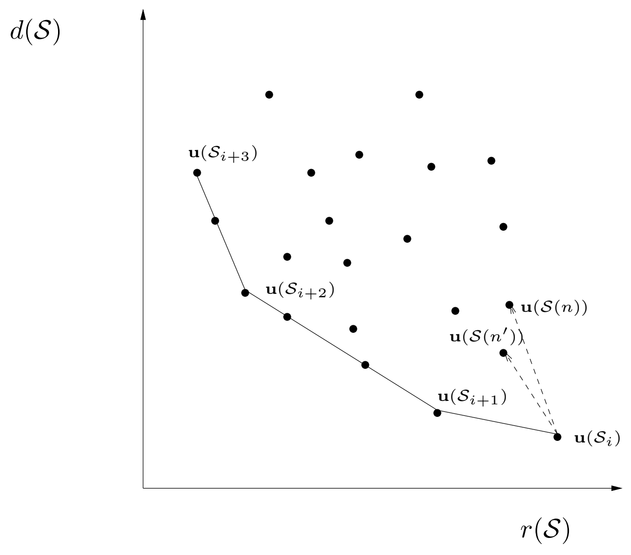
Let $\mathbf{u}(\mathcal{S}) = (r(\mathcal{S}), d(\mathcal{S}))$ denote the point with coordinates $r(\mathcal{S})$ and $d(\mathcal{S})$. Let H denote the lower boundary of the convex hull of the set of all points $\mathbf{u}(\mathcal{S})$ corresponding to pruned subtrees $\mathcal{S} \preceq \mathcal{T}$. Then any point $\mathbf{u}(\mathcal{S})$ on H corresponds to a pruned subtree that is a solution to the optimization problem.

The generalized BFOS algorithm finds the optimal subtrees associated to the extreme points (vertices) $\mathbf{u}(\mathcal{S}_i),\ i = 1, \ldots, m$ of the lower boundary of the convex hull using the following two results (proofs are in [5]):

1. These optimal pruned subtrees are nested, that is, $\mathcal{S}_m \preceq \cdots \preceq \mathcal{S}_2 \preceq \mathcal{S}_1$.

2. Any pruned subtree $\mathcal{S}_{i+1}$ can be obtained from $\mathcal{S}_i$ by successively pruning *one single* branch and the interim pruned subtrees correspond to points on the segment connecting $\mathbf{u}(\mathcal{S}_i)$ and $\mathbf{u}(\mathcal{S}_{i+1})$. Thus the interim pruned subtrees are also optimal. Pruning is done at the interior node n that minimizes the magnitude of the slope of vector $\mathbf{u}(\mathcal{S}(n))\mathbf{u}(\mathcal{S}_i)$. Here $\mathcal{S}(n)$ is a pruned subtree of $\mathcal{S}_i$ obtained by pruning one single branch rooted at n (see Figure 1).

Due to the monotonicity assumption we have $\mathcal{S}_m = n_0$ and $\mathcal{S}_1 = \mathcal{T}$. Thus, one starts from the whole tree $\mathcal{T}$ and proceeds by successively pruning one branch until n_0 is reached. A pseudocode of the generalized BFOS algorithm using an efficient data structure is given in [12].



$d(\mathcal{S})$
$\mathbf{u}(\mathcal{S}_{i+3})$
$\mathbf{u}(\mathcal{S}_{i+2})$
$\mathbf{u}(\mathcal{S}(n))$
$\mathbf{u}(\mathcal{S}(n'))$
$\mathbf{u}(\mathcal{S}_{i+1})$
$\mathbf{u}(\mathcal{S}_{i})$
$r(\mathcal{S})$

Figure 1. Optimal tree pruning.

3 Implementation of optimal tree pruning for fractal image compression

The generalized BFOS algorithm has been successfully used in many applications including tree-structured vector quantization [5] and optimal bit allocation. In [25], we give the first application of the generalized BFOS algorithm to fractal image compression. The method presented there proceeds as follows.

- Step 1. Top-down heuristic hierarchical partitioning without searching.
- Step 2. Bottom-up merging of the ranges in optimal rate-distortion sense.

Given a range block corresponding to an inner node in the partition tree there are many different ways to split the block into two or more subblocks corresponding to the child nodes of the given inner node. The quadtree scheme is one of the simplest possible ways. We use the optimal pruning algorithm with hierarchical partitions consisting of rectangles (HV partitions, [10]). This choice is motivated by three reasons:

- 1. Rectangular partitions lead to better results than quadtrees, both in terms of rate-distortion performance and visual quality [10].
- 2. Generating the initial fine grained partition is computationally simple.

- 3. It allows a comparison with one of the best pure (i.e., non-hybrid) fractal coding schemes, the HV coder [10] of Fisher.

3.1 The initial HV partition

In an HV partition a rectangular range block can be split either horizontally or vertically into two smaller rectangles. A decision about the split location has to be made. Whereas [10] adopts a criterion based on edge location, we follow [30, 20] and propose to split a rectangle such that an approximation by its DC component[1] in each part gives a minimal total square error. We expect fractal coding to produce relatively small collage errors with this choice because

- approximation by the DC component alone (which is part of the fractal encoding) will already give small sums of squared errors by design of the splitting scheme, and
- for the approximation of the dynamic part of the range blocks we have more domains available, if the range block variances are low. This comes from the limitations of the scaling factor, $|s| \le s_{\max}$.

The details are as follows. We consider a digital image given by a real-valued function $f : \mathcal{R}_0 \to \mathbf{R}$, where $\mathcal{R}_0$ is a set of integer pixel coordinates (x, y) from a rectangular grid. For any range block R the set of pixel coordinates $\mathcal{R}$ can be written as

$$\mathcal{R} = \{x_{\min}, ..., x_{\max}\} \times \{y_{\min}, ..., y_{\max}\}.$$

We define the size $|\mathcal{R}|$, the mean $\mu(\mathcal{R})$, and the square error $E(\mathcal{R})$ with respect to DC approximation:

$$\begin{aligned} |\mathcal{R}| &= (x_{\max} - x_{\min} + 1) \cdot (y_{\max} - y_{\min} + 1) \\ \mu(\mathcal{R}) &= \frac{1}{|\mathcal{R}|} \sum_{(x,y)\in\mathcal{R}} f(x, y) \\ E(\mathcal{R}) &= \sum_{(x,y)\in\mathcal{R}} (f(x, y) - \mu(\mathcal{R}))^2 \end{aligned}$$

We define the vertical split of a rectangle $\mathcal{R}$ into

$$\begin{aligned} \mathcal{R}_{\text{left}}(x_{\text{split}}) &= \{x_{\min}, ..., x_{\text{split}}\} \times \{y_{\min}, ..., y_{\max}\}, \\ \mathcal{R}_{\text{right}}(x_{\text{split}}) &= \{x_{\text{split}} + 1, ..., x_{\max}\} \times \{y_{\min}, ..., y_{\max}\} \end{aligned}$$

such that the split position x_{split} minimizes $E_V(x)$ the square error with respect to DC approximation , that is,

$$x_{\text{split}} = \arg \min_{x=x_{\min},\ldots,x_{\max}-1} E_V(x),$$

[1]The DC component of a block is defined here as the block whose pixel values are equal to the average intensity of the block.

where

$$E_V(x) = E(\mathcal{R}_{\text{left}}(x)) + E(\mathcal{R}_{\text{right}}(x)).$$

In the same way we define a horizontal split at $y = y_{\text{split}}$ with minimal square error $E_H(y_{\text{split}})$. The rectangle $\mathcal{R}$ will be split vertically at $x = x_{\text{split}}$, if $E_V(x_{\text{split}}) \leq E_H(y_{\text{split}})$. Otherwise $\mathcal{R}$ will be split horizontally at $y = y_{\text{split}}$.

We also prescribe a minimal horizontal and vertical rectangle size of 2 pixels and proceed to recursively divide blocks as long as possible. At the end we arrive at a hierarchical partition in which all leaf nodes of the partition tree correspond to range blocks of size 2 × 2, 2 × 3, 3 × 2, and 3 × 3 pixels.

In addition one can multiply the square errors $E_V(x)$ and $E_H(x)$ with a bias function $B(x)$ to penalize a split that results in very thin or very flat rectangles. We use a factor $0.4t^2 + 1$, where t goes from -1 to 1 as x runs from $x_{\min}$ to $x_{\max} - 1$, respectively, y runs from $y_{\min}$ to $y_{\max} - 1$.

3.2 Encoding a range block

Given a range block R_i we wish to approximate it as $R_i \approx sC + o\mathbf{1}$ where $C = AD$ is a *codebook block* of the same size. The codebook blocks are provided in a very simple way:

- by pixel averaging we downsample the original image by a factor of 2, and
- for a given range block R_i we define the corresponding codebook blocks as all vectors obtained from all possible blocks of the same size in the downsampled image.

We assume that the codewords of respectively all domain blocks, all scaling factors and all offsets are of the same length. Thus, as explained in Section 1 almost optimal parameters are found by least squares optimization. This requires for all range blocks R_i in the hierarchical partition (leaf nodes *and* inner nodes) and all corresponding codebook blocks C the computation of the inner products $\langle R_i, \mathbf{1}\rangle$, $\langle R_i, R_i\rangle$, $\langle R_i, C\rangle$, $\langle C, \mathbf{1}\rangle$, $\langle C, C\rangle$ (see [24]). These computations can be done efficiently by using the hierarchical ordering of the range blocks. We save some of the inner products in arrays and reuse them for rapid inner product calculation for the range block corresponding to the parent node in the hierarchy. To achieve this with the minimum overhead of memory usage the computation should be organized recursively so that temporary storage for arrays of inner products is necessary only for a sequence of nodes on a single path from the root node to a leaf node of the partitioning tree.

In our implementation we use a very simple bit allocation scheme for a given range block, explained here using the example of a 512 × 512 grey scale image. In this case the downsampled image is of size 256 × 256 and storing a domain block address $(x_{\min}, y_{\min})$ costs 8 bits for the x- and y-components each. The quantization of the offset o proceeds along the improved method devised in [14]. Overall, we have the bit allocation

- 5 bits for the quantized scaling coefficient $\overline{s}$,
- 6 bits for the quantized offset $\overline{o}$,
- 16 bits for the domain block address.

If $\overline{s} = 0$, then the range is approximated as a DC block and no domain address is necessary. Thus, either 11 or 27 bits are used per range block.

3.3 Optimal tree pruning

For the optimal tree pruning with the generalized BFOS algorithm we define two tree functionals for our application, the rate $r(\cdot)$ and the distortion $d(\cdot)$ as follows.

The rate must include the bits for the transformation lists and for the partition. We can store the HV partition using a tree traversal. At each node we specify with one bit whether it is an inner node or a leaf node. For an inner node n we also must specify the way the corresponding rectangle is split, using $r_{\text{split}}(n)$ bits,

$$r_{\text{split}}(n) = \begin{cases} 1 + \lceil \log_2(\text{width}(n) - 3) \rceil & \text{vertical split}, \\ 1 + \lceil \log_2(\text{height}(n) - 3) \rceil & \text{horizontal split}. \end{cases}$$

The first bit indicates a horizontal or vertical split whereas the remaining bits are used to specify the split location relative to the upper left corner of the rectangle. Note that blocks have width and height of at least 2 pixels.

We are now ready to apply the setting of the generalized BFOS algorithm as explained in Section 2. Let $\mathcal{T}$ denote the initial partitioning tree as obtained in Section 3.1. The rate associated with a node $n \in \mathcal{T}$ is

$$r(n) = r_{\text{code}}(n) + r_{\text{side}}(n)$$

where $r_{\text{code}}(n) = 27$ bits (11 bits, if the scaling factor $s = 0$) and the partition information $r_{\text{side}}(n)$ of the tree, charged to the node n, is defined recursively by

$$r_{\text{side}}(n) = \begin{cases} 1, \text{ if } n \text{ is the root node}, \\ 1 + \frac{1}{2}(r_{\text{side}}(n.p) + r_{\text{split}}(n.p)), \text{ otherwise}, \end{cases}$$

where $n.p \in \mathcal{T}$ denotes the parent node of $n \in \mathcal{T}$. Then, for any subtree $\mathcal{S} \preceq \mathcal{T}$ we have that the total rate

$$r(\mathcal{S}) = \sum_{n \in \tilde{\mathcal{S}}} r(n)$$

precisely indicates the number of bits needed to store the corresponding partition and the transformation lists.

The distortion at node $n \in \mathcal{T}$ is the least squares collage error

$$d(n) = \min_k E_c(R(n), D_k),$$

where $R(n)$ denotes the range corresponding to node n. For any subtree $\mathcal{S} \preceq \mathcal{T}$ the total distortion

$$d(\mathcal{S}) = \sum_{n \in \tilde{\mathcal{S}}} d(n)$$

is the overall collage error.

To apply the generalized BFOS algorithm we must ensure that our tree functionals are linear and monotonic. Both rate and distortion are linear already by construction. Moreover, it can be shown that they are also monotonic. This, however, may necessitate a technical modification at those few nodes of the tree where both child ranges, encoded with only 11 bits as DC blocks, require a combined rate that is less than that of the parent node where 27 bits are used for the transformation list. Therefore, the theory is applicable, and optimal subtrees can be extracted with the generalized BFOS algorithm. As a result the corresponding fractal codes are optimal.

3.4 Greedy tree growing

As an alternative to the optimal hierarchical partitions we consider faster, but suboptimal, greedy tree growing strategies. They produce hierarchical HV partitions with a user given number of range blocks. Two parameters are used:

- a small tolerance for some block error estimate, τ,
- a minimal linear block size, $l_{\min}$.

Let $R(n)$ be a block at node n corresponding to an image rectangle $\mathcal{R}$. For the error estimate we use either the traditional collage error in the root-mean-square (rms) version

$$\sqrt{\min_k E_c(R(n), D_k)/|\mathcal{R}|}$$

or, much simpler and faster to compute, just the block variance

$$E(\mathcal{R})/|\mathcal{R}|$$

of the corresponding rectangle $\mathcal{R}$.

The procedure operates with a maximum heap of image rectangles $\mathcal{R}$. The heap is sorted with respect to rms collage error or block variance. Then we iteratively extract rectangles $\mathcal{R}$ from the maximum heap and process them as follows. Depending on the choice of the type of greedy method (rms-error based or variance based), if either

$$\sqrt{\min_k E_c(R(n), D_k)/|\mathcal{R}|} \geq \tau \quad \text{or} \quad E(\mathcal{R})/|\mathcal{R}| \geq \tau$$

and

$$\max(\text{width}(\mathcal{R}), \text{height}(\mathcal{R})) \geq 2 l_{\min},$$

number ranges	comp. ratio	partition information (bits)	list code size (bits)	PSNR (dB)
10000	6.47	54865	269071	39.10
5000	12.65	31750	134004	36.07
4000	15.70	26571	107027	35.12
3000	20.76	20976	80022	33.89
2000	31.00	14889	52751	32.13
1000	61.43	8195	25942	29.43
500	123.78	4412	12531	27.05

Table 1. Encoding results with the optimal tree pruning technique for the 512 $\times$ 512 Lenna image.

then we subdivide the rectangle as in Section 3.1 and insert the two subrectangles into the heap. Otherwise, we output rectangle $\mathcal{R}$. If the sum of the number of rectangles already output and the number of rectangles that are still in the heap is equal to the number of desired ranges, we output all rectangles in the heap and end the partitioning procedure.

4 Results

In this section we show the encoding results for the 512 $\times$ 512 (8 bpp) Lenna image. For the rate-distortion optimal encodings Table 1 lists the compression ratios, the way the bits are distributed between partition code and list code, and the peak-signal-to-noise ratios. In Figure 2 we plot the corresponding rate-distortion curve along with the ones obtained using the greedy methods discussed in Section 3.4. With quadtree partitions the simple method based on block variance produces partitions that yield the same quality encodings as the standard top-down collage rms-error based approach [26]. For the hierarchical HV partitions, however, the greedy collage error based partitioning method is better than the simpler one using the block variance threshold. But the method based on the BFOS tree pruning algorithm outperforms both of them. Of course, this is to be expected due to the proven optimality of the algorithm.

The HV coder of Fisher and Menlove in [10] uses the greedy collage error based partitioning approach (not with the same split strategy though). Their best result, obtained by allowing more codebook blocks than in our experiment and with entropy coding of the scaling factor and offset indices, is of the same quality as the top curve in our graph.

Because we did not use complexity reduction techniques, the encoding time in the experiment reported here is large (several hours for the entire rate-distortion curve).

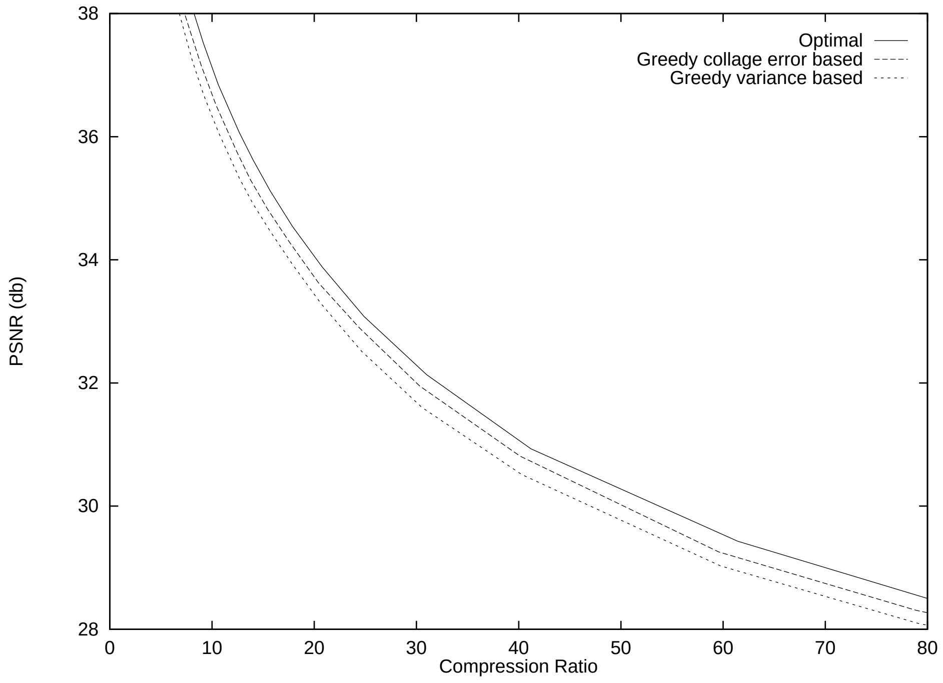


Figure 2. Rate distortion curve for 512 × 512 image Lenna.

5 Extensions

Three further improvements of this work are currently being investigated.

5.1 Entropy coding of the partition information.

The partition code can be made shorter by considering contextual entropy coding. The bit which distinguishes inner nodes from leaf nodes, for example, is highly dependent on the corresponding range block size. The bit specifying a horizontal or vertical split depends to some extent on the aspect ratio of the range block (a vertical split is more likely for a wide range, a horizontal split more likely for a tall range). Also the split location for a given range size has a strongly non-uniform probability distribution which can be exploited by entropy coding. Most of these modifications can be integrated, so that the reduced bit rates are observed by the optimization technique. First tests indicate that with such context-based adaptive arithmetic coding the information for the partition can be reduced to about 80% of the sizes reported in the table.

5.2 Variable length coding.

Instead of fixed length encoding for the coefficients s, o and the domain block address, variable length codes can be considered. The design of these codes can be organized so that an implementation within the concept of optimal tree pruning is possible. The trees then may contain sequences of unary nodes corresponding to range block encodings with an increasing number of bits. (In fact, in the implementation used here we already allow a crude variant of such variable length coding of a range block by considering coding it the regular way, i.e., as $\overline{s}AD + \overline{o}\mathbf{1}$, as well as a constant intensity block, i.e., with $\overline{s} = 0$ which saves some bits for the domain block address.)

5.3 Other adaptive initial partitions.

The splitting criterion for the rectangular partitions, considered in [25] is only one possible heuristic. Other splitting strategies as well as more adaptive partitions can be tested. For example, the polygonal partition [30, 20] in which the rectangular one is extended to allow also a split in the diagonal or anti-diagonal direction is straightforward to implement in the context of the generalized BFOS algorithm.

6 Other rate-distortion based schemes

Most of the fractal image compression schemes proposed in the literature use heuristic techniques and do not guarantee optimal codes. In this section we review works that rely on a rate-distortion criterion.

In [2] and for uniform partitions, it is observed that in many cases best domain candidates are located in the neighborhood of the range blocks. Thus, rate-distortion gains can be achieved by allowing domain blocks to have variable length codewords. For each range block a codebook of domain blocks is designed such that domain blocks nearest to the range block have shortest codewords. With this setting, for a given bit budget r_0, optimal domain blocks in problem (1.2) are not necessarily those that minimize the collage error. Therefore, it is suggested to select the domain blocks as follows. The encoder starts with a fractal code where all range blocks are encoded at lowest rate, that is, domain blocks in each transformation list have shortest codewords. Then, the objective function d is reduced iteratively by allocating at each step more bits to the domain block of one of the range blocks. This range block R_k is one that maximizes the reduction in collage error per additional domain block bit, that is,

$$k = \arg \max_{i=1,\ldots,n_R} \lambda_i$$

where

$$\lambda_i = \max_m \frac{E_c(R_i, D) - E_c(R_i, D_m)}{r_{D_m} - r_D}.$$

Here D is the domain block in the current transformation list of R_i, and D_m is a domain block whose codeword size r_{D_m} is strictly larger than r_D, the codeword size of D. The iteration is stopped when the total bit rate of the code becomes larger than the bit budget r_0. One can prove that this algorithm finds an optimal solution to the constrained problem (see [29]).

Lu [18] uses an algorithm, which finds an optimal solution to problem (1.2) in the context of quadtree partitions (hierarchical partition where all ranges are square blocks) and domain blocks of variable length codewords. The algorithm, which was previously proposed in [27] for quadtree-based vector quantization, relies on a Lagrange multiplier method and consists of solving the unconstrained optimization problem

$$\min_{W \in \mathcal{W}} d(W) + \lambda l(c(W)) \tag{6.1}$$

where $\lambda \in [0, \infty)$. Indeed, one can show [7] that for any choice of $\lambda \geq 0$ a solution W^* to the unconstrained problem (6.1) is a solution to the constrained problem

$$\min_{W \in \mathcal{W}} d(W)$$

subject to

$$l(c(W)) \leq l(c(W^*)).$$

In the quadtree scheme, a (parent) square block is either accepted in the partition or subdivided into four smaller (child) square blocks. A more adaptive scheme [11] can be obtained by allowing 10 other intermediary configurations (see Figure 3). The authors use the Lagrange multiplier technique to determine the optimal adaptive partition corresponding to a three-level hierarchy.

Finally, the performance of fractal image coding is improved by hybrid schemes which incorporate other coding techniques, e.g., Discrete Cosine Transform or wavelet-based. These hybrid schemes are particularly efficient when a rate-distortion criterion is used [2, 13, 28, 19, 17].

7 Conclusion

We have presented a framework for creating rate-distortion optimal hierarchical partitions for fractal image compression. Even though our implementation based on rectangular HV partitions is crude, we reach the state-of-the art results of [10]. We regard our work as a contribution to a general solution of the optimal bit allocation problem in fractal image compression. The goal of these efforts is to allow variable bit rates not only considering many possible range block sizes but also by applying variable length codes for all components of the transformation parameters (s, o, domain address) of each range block. Even though the number of such possible fractal image codings is huge, a rate-distortion optimal encoding can be derived efficiently by the generalized BFOS algorithm.

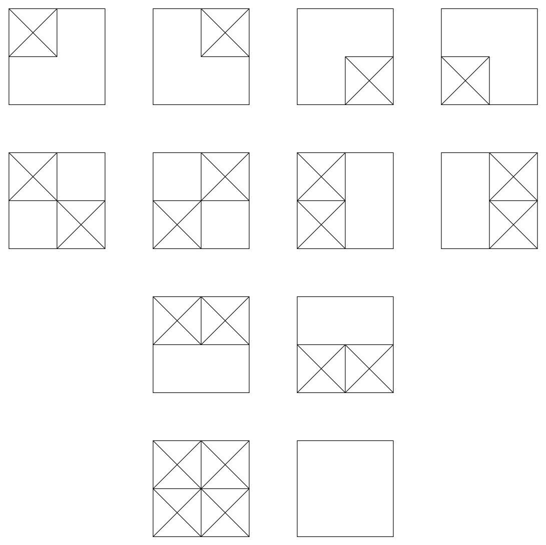


Figure 3. Admissible block partitions. The parent block is the large square and the child blocks are the smaller blocks inside marked with a cross.

Bibliography

1. Barnsley, M., *Fractals Everywhere,* Academic Press, San Diego, 1988.

2. Barthel, K. U., Schüttemeyer, J., Voyé, T., Noll, P., *A new image coding technique unifying fractal and transform coding,* in: *Proc. ICIP-94 IEEE International Conference on Image Processing,* Vol. III, pp. 112–116, Austin, Texas, Nov. 1994.

3. Bedford, T., Dekking, F. M., Breewer, M., Keane, M. S., van Schooneveld, D., *Fractal coding of monochrome images,* Signal Processing: Image Communication 6 (1994) 405–419.

4. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., *Classification and Regression Trees,* Wadsworth, Belmont, California, 1984.

5. Chou, P. A., Lookabaugh, T., Gray, R. M., *Optimal pruning with applications*

to tree-structered source coding and modeling, IEEE Trans. Inform. Theory 35 (1989) 299–315.

6. Davoine, F., Robert, G., Chassery, J.-M., *How to improve pixel-based fractal image coding with adaptive partitions,* in: *Fractals in Engineering,* J. L. Vehel, E. Lutton and C. Tricot (eds.), Springer Verlag, London, 1997.

7. Everett, H., *Generalized Lagrange multiplier method for solving problems of optimum allocation of resources,* Operations Research 11 (1963) 399–417.

8. Fisher, Y. (ed.), *Fractal Image Compression — Theory and Application,* Springer-Verlag, New York, 1994.

9. Fisher, Y., Jacobs, E. W., Boss, R. D., *Fractal image compression using iterated transforms,* in: *Image and Text Compression,* J. A. Storer (ed.), Kluwer Academic Publishers, Boston, 1992.

10. Fisher, Y., Menlove, S., *Fractal encoding with HV partitions,* in: *Fractal Image Compression — Theory and Application,* Y. Fisher (ed.), Springer-Verlag, New York, 1994.

11. Fuchigami, T., Yano, S., Komatsu, T., Saito, T., *Fractal block coding with cost-based image partitioning,* in: *Proc. International Picture Coding Symposium PCS'96,* pp. 335–340, Melbourne, March 1996.

12. Gersho, A., Gray, R.M., *Vector Quantization and Signal Compression,* Kluwer, Boston, 1992.

13. Gharavi-Alkhansari, M., Huang, T., *Fractal image coding using rate-distortion optimized matching pursuit,* in: *Proc. SPIE Conf. Visual Communications and Image Processing,* pp. 1386–1393, Vol. 2727, Orlando, Florida, March 1996.

14. Hartenstein, H., Saupe, D., Barthel, K. U., *VQ-encoding of luminance parameters in fractal coding schemes,* in: *Proc. ICASSP'97 IEEE Int. Conf. Acoustics Speech and Signal Processing,* Vol. IV, pp. 2701–2704, Munich, April 1997.

15. Jackson D. J., Mahmoud W., Stapleton W., Gaughan P. T., *Faster fractal image compression using quadtree recomposition*, Image and Vision Computing (15)10 (1997) 759–767.

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

17. Li, J., Kuo, C.-C. J., *Fractal wavelet coding using a rate-distortion constraint,* in: *Proc. ICIP-96, IEEE International Conference on Image Processing,* Vol. II, pp. 81-84, Lausanne, Sept. 1996.

18. Lu, N., *Fractal Imaging,* Academic Press, 1997.

19. Melnikov, G., Katsaggelos, A., *A non uniform segmentation optimal hybrid fractal/DCT image compression algorithm,* in: *Proc. ICASSP'98 IEEE Int. Conf. on Acoustics, Speech and Signal Processing,* Seattle, Washington, May 1998.

20. Reusens, E., *Partitioning complexity issue for iterated function systems based image coding,* in: *Proc. EUSIPCO'94 VIIth European Signal Processing Conference,* Vol. I, pp. 171–174, Edinburgh, Sept. 1994.

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

22. Ruhl, M., Hartenstein, H., Saupe, D., *Adaptive partitionings for fractal image compression,* in: *Proc. ICIP-97 IEEE International Conference on Image Processing,* Vol. II, pp. 310–313, Santa Barbara, California, Oct. 1997.

23. Saupe, D., *Lean domain pools for fractal image compression,* in: *Proc. IS&T/SPIE 1996 Symposium on Electronic Imaging: Science & Technology – Still Image Compression II*, Vol. 2669, pp. 150–157, San Jose, California, Jan. 1996.

24. Saupe, D., Hartenstein, H., *Lossless acceleration of fractal image compression by fast convolution,* in: *Proc. ICIP-96 IEEE International Conference on Image Processing*, Vol. I, pp. 185-188, Lausanne, Sept. 1996.

25. Saupe, D., Ruhl, M., Hamzaoui, R., Grandi, L., Marini, D., *Optimal hierarchical partitions for fractal image compression,* in: *Proc. ICIP-98 IEEE International Conference on Image Processing,* Chicago, Oct. 1998.

26. Saupe, D., Jacob, S., *Variance-based quadtrees in fractal image compression,* Electronics Letters 33,1 (1997) 46–48.

27. Sullivan, G. J., Baker, R. L., *Efficient quadtree coding of images and video,* IEEE Trans. on Image Processing 3,3 (1994) 327–331.

28. Wakefield, P., Bethel, D., Monro, D., *Hybrid image compression with implicit fractal terms,* in: *Proc. ICASSP'1997 IEEE International Conference on Acoustics, Speech and Signal Processing,* Vol. IV, pp. 2933–2936, Munich, April 1997.

29. Westerink, P. H., Biemond, J., Boekee, D. E., *An optimal bit allocation algorithm for sub-band coding,* in: *Proc. ICASSP'88 IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 757–760, 1988.

30. Wu, X., Yao, C., *Image coding by adaptive tree-structured segmentation,* in: *Proc. DCC'91 Data Compression Conference,* J. A. Storer and M. Cohn (eds.), IEEE Comp. Soc. Press, 1991.