OPTIMAL HIERARCHICAL PARTITIONS FOR FRACTAL IMAGE COMPRESSION

Dietmar Saupe[1], *Matthias Ruhl*[2], *Raouf Hamzaoui*[2], *Luigi Grandi*[3], *Daniele Marini*[3]

[1]Universität Leipzig, Institut für Informatik, Augustusplatz 10–11, 04109 Leipzig, Germany
[2]Universität Freiburg, Institut für Informatik, Am Flughafen 17, 79110 Freiburg, Germany
[3]Universita' degli Studi di Milano, Dipartimento di Scienze della Informazione, 20135 Milano, Italy

ABSTRACT

In fractal image compression a partitioning of the image is required. In this paper we discuss the construction of rate-distortion optimal partitions. We begin with a fine scale partition which gives a fractal encoding with a high bit rate and a low distortion. The partition is hierarchical, thus, corresponds to a tree. We employ a pruning strategy based on the generalized BFOS algorithm. It extracts subtrees corresponding to partitions and fractal encodings which are optimal in the rate-distortion sense. First results are included for the case of fractal encodings based on rectangular (HV) partitions. We also provide a comparison with greedy partitions based on the traditional collage error criterion or just using block variance.

1. INTRODUCTION

One of the weaknesses of fractal image compression [1, 2] is its lack of good bit allocation procedures. This issue is complex since not only may each image block be encoded using a variable number of bits, but the blocks may also be of different sizes. On top of this there is side information specifying the underlying image partition, which also must be taken into account. This paper provides a step towards a general solution for fractal image compression with hierarchical image partitions. Our work is a novel application of the generalized BFOS algorithm for optimal tree pruning.

This paragraph reviews the standard type of fractal image encoding and introduces the notation used in this work. An image is partitioned into several blocks, called range blocks. For a range block R we consider a pool of domain blocks twice the linear size. The domain blocks are shrunken by pixel averaging to match the range block size. This gives a pool of codebook blocks $D_1, \ldots, D_{N_D}$. For range R and codebook block D we let

$$(s, o) = \arg \min_{s,o \in \mathbf{R}} \|R - (sD + o\mathbf{1})\|^2$$

where $\mathbf{1}$ is the flat block with unit intensity at every pixel. The coefficient s is clamped to $[-s_{\max}, s_{\max}]$ with $s_{\max} < 1$ to ensure convergence in the decoding and then both s and o are uniformly quantized yielding $\overline{s}$ and $\overline{o}$. The collage error for range R and codebook block D is $E_c(D,R) = \|R - (\overline{s}D + \overline{o}\mathbf{1})\|^2$. Finding a codebook block D_k with minimal collage error $E_c(D_k, R)$ yields an index k. The fractal code for range R consists of this optimal index k (the domain block address) and the corresponding quantized scaling and offset parameters $\overline{s}$ and $\overline{o}$.

Finding the optimal partition at a given bit rate is a difficult problem in fractal image compression since the space of all partitions with a given number of ranges is too large to allow to investigate all of them. Instead, heuristic methods must be developed in order to arrive at a suboptimal solution. A number of subdivision types have been used: uniform, quadtree, rectangular, triangular, other polygonal, and irregular partitions.

Top-down versus bottom-up. Most of the methods can be distinguished by the way the partition is generated. 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. 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 to arrive at a collection of partitions of varying complexity.

Searching versus no searching. In addition we may classify methods with respect to whether fractal codes are generated already during the partitioning process or only afterwards. One common approach is to subdivide a range block only if an estimate of the local approximation error for the range (i.e., the collage error) is above some threshold. In contrast, 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.

Of course, combinations of these categories are possible. For example, one may design a quadtree partition in the top-down fashion with or without searching, and then start to merge pairs of neighboring range blocks.

Results of previous research [3] indicate that it is generally not advantageous in terms of rate-distortion performance to create the partition without searching. However, in our work [4] we come to a somewhat different conclusion. In [4] we showed for the special case of quadtree partitions that one can indeed base the decision whether or not to split a square entirely on image statistics (size, mean, and variance) without sacrificing rate-distortion quality.

The method presented in this paper 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.

The tool to achieve rate-distortion optimal partitions is the generalized BFOS algorithm, which we review in the following section. In Section 3 we apply the method to fractal image coding using rectangular (HV) partitions. An initial fine grained partition is heuristically derived in Subsection 3.1, the fractal coding is explained in Subsection 3.2, while the following subsection makes the connection with the BFOS algorithm with which we achieve optimal pruned HV partitions. The last subsection presents two greedy suboptimal partitioning strategies. Section 4 contains simulation results, and in the remainder of the paper we discuss extensions of the method and conclusions.

2. OPTIMAL TREE PRUNING USING THE GENERALIZED BFOS ALGORITHM

The terminology and notation used in this section closely follow those in [5] 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 $\mathcal{T}$.

The problem of optimal tree pruning consists of finding the list of pruned subtrees $\mathcal{S}^* \preceq \mathcal{T}$ which minimize the Lagrangian cost functional

$$u_2(\mathcal{S}) + \lambda u_1(\mathcal{S})$$

where u_1 and u_2 denote two tree functionals (such as rate and distortion) and λ increases from 0 to ∞. Thus, any solution $\mathcal{S}^*$ minimizes the tree functional u_2 over all pruned subtrees of $\mathcal{T}$ having the same or lower tree functional value $u_1(\mathcal{S}^*)$. When $\mathcal{T}$ has a large number of nodes, solution by full search is impractical. Fortunately, if we assume that the tree functionals u_1 and u_2 are monotonic and linear, with u_1 monotonically increasing and u_2 monotonically decreasing, then the problem can be efficiently treated by the algorithm proposed by Chou *et al.* [6, 5], which is a generalization of the BFOS algorithm of Breiman, Friedman, Olshen, and Stone [7]. Here a tree functional u is said to be monotonically increasing (resp. monotonically decreasing) if

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

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 $\mathcal{S}$.

Observe first that if $\mathbf{u}(\mathcal{S}) = (u_1(\mathcal{S}), u_2(\mathcal{S}))$ denotes the point with coordinates $u_1(\mathcal{S})$ and $u_2(\mathcal{S})$, then the problem of optimal tree pruning is equivalent to finding the points $\mathbf{u}(\mathcal{S}^*)$ on the lower boundary of the convex hull of the set of all points corresponding to the pruned subtrees $\mathcal{S} \preceq \mathcal{T}$. 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 [6]):

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.

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 [5].

The generalized BFOS algorithm has been successfully used in many applications including tree-structured vector quantization [6] and optimal bit allocation. In this paper, we give the first application to fractal image compression.

3. IMPLEMENTATION OF OPTIMAL TREE PRUNING FOR FRACTAL IMAGE COMPRESSION

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. Here we propose as an example of the optimal pruning algorithm to use hierarchical partitions consisting of rectangles (HV partitions, [8]). This choice is motivated by three reasons:

1. Rectangular partitions have been shown to lead to better results than quadtrees, both in terms of rate-distortion performance and visual quality [8].
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 [8] 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. While [8] adopts a criterion based on edge location we follow [9, 10] 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 a larger part of the domain available, if the range block variances are low. This comes from the limitations of the scaling factor, $|s| \leq s_{\max}$.

The details are as follows. We consider a digital image given by a real-valued function $f : \mathcal{R}_0 \rightarrow \mathbf{R}$, 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 $E(\mathcal{R})$ w.r.t. DC approximation:

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

[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.

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 the square error $E_V(x)$ w.r.t. DC approximation, i.e.,

$$\begin{aligned}x_{\text{split}} &= \arg \min_{x=x_{\min},\dots,x_{\max}-1} E_V(x), \\ E_V(x) &= E(\mathcal{R}_{\text{left}}(x)) + E(\mathcal{R}_{\text{right}}(x)).\end{aligned}$$

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 $B(x)$ in order to penalize a split that results in very thin or very flat rectangles. We use a function $\beta(t^2 + 1)$, where t goes from -1 to 1 as x runs from $x_{\min}$ to $x_{\max} - 1$, and $\beta = 0.4$.

3.2. Encoding a range block

Given a range block R we wish to approximate it by $R \approx sD + o\mathbf{1}$ where D is a codebook block of the same size. Our codebooks 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 we define the corresponding codebook as the set of all vectors obtained from all possible blocks of the same size in the downsampled image.

For all range blocks R in the hierarchical partition (leaf nodes *and* inner nodes) and all corresponding codebook blocks D we need to calculate the inner products $\langle R, \mathbf{1}\rangle$, $\langle R, R\rangle$, $\langle R, D\rangle$, $\langle D, \mathbf{1}\rangle$, $\langle D, D\rangle$ in order to compute the optimal (quantized) coefficients $\overline{s}$, $\overline{o}$, and the collage error $E_c(D, R)$, see [11]. These computations can be made efficient by making use of 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 first implementation presented here 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 [12]. Overall, we have the bit allocation

- 5 bits for the scaling coefficient $\overline{s}$,
- 6 bits for the 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. For simplicity we do not use isometries in our fractal coder.

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 $r(\cdot) = u_1(\cdot)$ and the distortion $d(\cdot) = u_2(\cdot)$ as follows.

The rate must include the bits for the block codes and the side information 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 while 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 side 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 fractal code.

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

$$d(n) = E_c(R(n), D(n))$$

where $(R(n), D(n))$ denotes the range-domain pair corresponding to the 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.

In order 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.[2] Therefore, the theory is applicable, and optimal subtrees can be extracted with the generalized BFOS algorithm. As a result the corresponding partitions are rate-distortion optimal for fractal encoding.

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, $l_{\min}$.

For the error estimate we use either the traditional collage error for block R corresponding to the image rectangle $\mathcal{R}$ in the root-mean-square version

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

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 w.r.t. 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

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

and

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

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 terminate the partitioning procedure.

4. RESULTS

In this summary we only show the encoding results for one image, 512 × 512 Lenna, thus providing a proof of concept. For the rate-distortion optimal encodings Table 1 below lists the compression ratios, the way the bits are distributed between partition code and transformation code, and the peak-signal-to-noise ratios. In figure 1 we plot the corresponding rate-distortion curve along with the ones obtained using the greedy methods above. When using the quadtree partitions we found earlier in [4] that the simpler method based on block variance produces partitions that yield the same quality encodings as the standard collage rms-error based approach. For the hierarchical HV partitions, however, we see from the results in this study that the greedy collage error based partitioning method is better than the simpler one using the block variance threshold. However, the method developed in the paper, based on the BFOS tree pruning algorithm outperforms both of them. Of course, this outcome is to be expected due to the proven optimality of the algorithm.

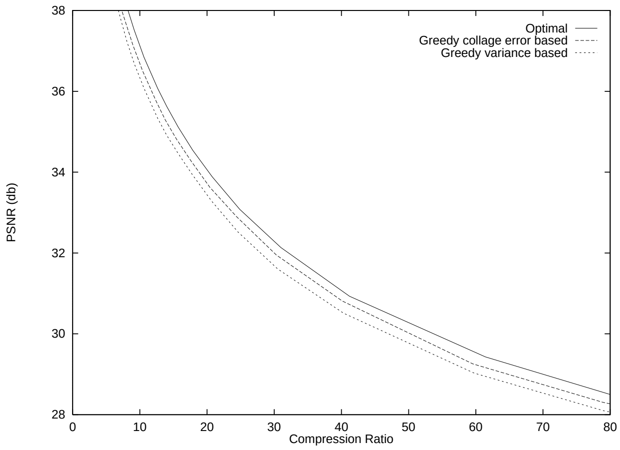

Figure 1: Rate distortion curve for 512 × 512 image Lenna.

number ranges	comp. ratio	side information (bits)	transf. 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 × 512 Lenna image.

The HV coder of Fisher and Menlove in [8] uses the greedy collage error based partitioning approach (not with the same split strategy though). Their best result, obtained with all options turned on, is of the same quality as the top curve in our graph. In our coder we have not yet employed any entropy coding and the domain pool searched is much smaller than the one used in the best result of [8].

The encoding time in the experiment reported here is large (several hours for the entire rate-distortion curve) since we have not made use of any complexity reduction techniques, which should be adapted to the special requirements set in this application.

5. EXTENSIONS

While this work introduces the application of an optimality criterion to the bit allocation for the partitioning, three further improvements are currently being investigated.

Entropy coding of the side information. All components of the side information can be made more efficient 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

[2] This 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 code.

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 side information can be reduced to about 80% of the sizes reported in the table above.

Variable length coding. Instead of fixed length encoding for range-domain transformation, i.e., 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 $\overline{s}D + \overline{o}\mathbf{1}$, as well as a constant intensity block, i.e., with $\overline{s} = 0$ which saves some bits for the domain block address.)

Other adaptive initial partitions. The splitting criterion for the rectangular partitions, considered in this paper is only one possible heuristic. Other splitting strategies as well as more adaptive partitions can be tested. For example, the polygonal partition [9, 10] 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 our optimal partitioning method described here.

6. CONCLUSION AND OUTLOOK

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 results of HV coding in [8]. 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 widely varying bit rates for the pixels of an image 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, an overall rate-distortion optimal bit allocation by means of the generalized BFOS algorithm is possible and should provide additional coding gains.

Acknowledgment. This work was supported in part by grant CSA95E4 of the European Commission. A part of this work was carried out by the first author at the Computer Engineering Department of the University of California, Santa Cruz. This support and fruitful discussions with Glen Langdon are gratefully acknowledged.

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

[3] 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.

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

[5] Gersho, A., Gray, R.M., *Vector quantization and signal compression,* Kluwer, Boston, 1992.

[6] 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.

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

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

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

[10] Reusens, E., *Partitioning complexity issue for iterated function systems based image coding,* in: *Proceedings of the VIIth European Signal Processing Conference EUSIPCO'94,* Edinburgh, Sept. 1994.

[11] Saupe, D., Hartenstein, H., *Lossless acceleration of fractal image compression by fast convolution,* IEEE International Conference on Image Processing (ICIP'96), Lausanne, Sept. 1996.

[12] Hartenstein, H., Saupe, D., Barthel, K. U., *VQ-encoding of luminance parameters in fractal coding schemes,* IEEE Intern. Conf. Acoustics Speech Signal Processing (ICASSP'96), München, April 1997.

[13] Saupe, D., *Lean domain pools for fractal image compression,* in: Proceedings from IS&T/SPIE 1996 Symposium on Electronic Imaging: Science & Technology – Still Image Compression II, Vol. 2669, Jan. 1996.