Complexity Reduction Methods for Fractal Image Compression

Dietmar Saupe and Raouf Hamzaoui[1]

Institut für Informatik
Universität Freiburg
Am Flughafen 17
79110 Freiburg
Germany

Abstract: Fractal image compression allows fast decoding but suffers from long encoding times. During the encoding a large number of sequential searches through a list of domains (portions of the image) are carried out while trying to find a best match for another image portion called range. In this article we review and extend the methods that have been developed to reduce the time complexity of this searching. Also we present a new taxonomy of the methods, provide an evaluation and propose two new techniques.

Keywords: Image compression, classification, clustering, nearest neighbors, time complexity, fractals, Hilbert curve.

1 Introduction

Fractal image compression began with the visionary conception of M. Barnsley in 1987 and the ground breaking work of A. Jacquin [1, 2] in 1989. Since then a lot of work on the topic has been undertaken which has given fractal image compression the power to become a serious competitor of the established compression techniques. Fractal image compression is explained in the books [3, 4] and a recent survey of the rapidly growing literature in this field is given in [5].

Now it is common understanding that fractal image compression can deliver a performance in terms of a rate-distortion curve that is favorably comparable to, e.g., JPEG and which has a definite edge at very high com-

[1]This paper appears in: I.M.A. Conf. Proc. on *Image Processing; Mathematical Methods and Applications,* Sept. 1994, J. M. Blackledge (ed.), Oxford University Press, 1996.

pression ratios. Moreover, fractal image compression provides fast decoding and the code is resolution independent. However, the method suffers from long encoding times, which is its major deficiency. During the encoding a large pool of image subsets, called domains, has to be searched repeatedly many times, which by far dominates all other computations in the encoding process. If the number of domains in the pool is N_D, then the time spent for each search is *linear* in N_D, $O(N_D)$. Several methods have been devised to reduce the time complexity of the encoding. They all work by some way of *feature extraction*. One can consider a finite set of discrete features, which are chosen either a priori fixed or adaptively, i.e., depending on the image. Alternatively, it has been proposed to use continuous features, either a single one or several simultaneously yielding feature vectors. Thus, we can introduce the following taxonomy of these approaches:

1. **Discrete features.**

   (a) **Classification.** In classification methods domains are grouped independently and online into predefined classes of domains [2, 6, 7, 8]. For a given range only the class of the range is searched for a matching domain.

   (b) **Adaptive clustering.** In clustering methods domains and ranges are grouped around cluster centers which depend on the image and, thus, on the set of ranges and domains [9, 10]. As in the classification method, only the class of the given range is searched for a matching domain.

2. **Continuous features.**

   (a) **1D functional methods.** The domains and ranges are assigned a suitable one-dimensional real value, then ordered in a corresponding linear array and for a given range only those domains are considered whose values are within some distance of the value of the given range [11].

   (b) **Feature vectors.** The domains and ranges are assigned a multi-dimensional feature vector and for a given range only a few domains that are closest in feature space are considered [12, 13, 14].

These approaches reduce the factor of proportionality in the $O(N_D)$ time complexity for a search in the domain pool, except for the method based on feature vectors which can be organized such that it yields an $O(\log N_D)$ complexity. All the above methods are designed with the goal of a large domain pool, which is aimed at good image quality without losing too much in terms of compression ratios. Some authors are opposed to this reasoning. They suggest instead to rigorously restrict the domain pool size by considering only domains that are geometrically very close to the range in the image [15, 16] (see also Monro's contribution in this volume). Thus, with this approach one can encode images very rapidly, however at a certain cost in terms of the image quality of the encodings.

The remainder of this article is organized as follows. In section 2 we present the mathematical notations used in the paper. The main part of this article is in section 3 and consists of the descriptions of the methods. Due to space limitations we cannot present the algorithms in their entirety. Readers intending an implementation can find the details in the corresponding references. In section 4 we provide a partial empirical evaluation of these methods and point out how these methods can be combined to yield further acceleration. In section 5 we propose two new techniques based on feature vectors: a clustering method in higher dimensions and a functional approach employing a Hilbert scan of the feature vector space.

2 Notations

For the discussion in the paper let us assume that an image is partitioned into non-overlapping square blocks of size $N \times N$ called *range blocks*. This is not a restriction since it will be clear how the principles described carry over to more general partitions.

We consider each range block as a vector R in the linear vector space $\mathbf{R}^n$ where $n = N \times N$. The conversion from a square subimage of side length N to a vector of length $n = N^2$ can be accomplished, e.g., by scanning the block line by line. Working with vectors in place of 2D-arrays simplifies the notation considerably without losing generality.

The *domain pool* is a collection of square blocks which are typically larger than the ranges and taken also from the image, called *domain blocks*. The domain pool is enlarged by including blocks obtained after applying the eight isometrical operators to the domain blocks (i.e., rotations and reflections). Finally, by pixel averaging, the size of these blocks is reduced to the size of a range block. The resulting blocks are called *codebook blocks*.

In the encoding process for a range block a search through the codebook blocks is required. A vector representing a codebook block will be denoted by D. A small set of $p < n$ blocks independent from the image is also considered. We represent them by the vectors $B_1, B_2, \ldots, B_p \in \mathbf{R}^n$, which are chosen so as to form an orthonormal basis of a p-dimensional subspace of $\mathbf{R}^n$. They are known as *the fixed basis blocks*. The encoding problem can then be stated as the least squares problem

$$\min_{x \in \mathbf{R}^{p+1}} ||R - Ax||, \tag{2.1}$$

where A is an n by $p+1$ matrix whose columns are $D, B_1, B_2, \ldots, B_p$ and

$x = (a, b_1, \ldots, b_p) \in \mathbf{R}^{p+1}$ is a vector of coefficients. This problem should be solved for all codebook blocks D and the one which gives the smallest error $||R - (aD + \sum_1^p b_k B_k)||$ is selected on condition that the value of the scale factor a for the codebook block D ensures the convergence of the decoding process (e.g., by requiring $|a| < 1$). A basic result of linear algebra states that if the codebook block D is not in the linear span of the fixed basis blocks $B_1, \ldots, B_p$, then the minimization problem (2.1) has a unique solution

$$\bar{x} = (A^T A)^{-1} A^T R \tag{2.2}$$

where the matrix $A^+ = (A^T A)^{-1} A^T$ is also known as the pseudo-inverse of A. Thus, the range block R is approximated by the *collage* block AA^+R where AA^+ is the orthogonal projection matrix onto range(A). Now let P be the orthogonal projection operator which projects $\mathbf{R}^n$ onto the subspace $\mathcal{B}$ spanned by only the fixed basis blocks $B_1, B_2, \ldots, B_p$. Thus, by orthogonality of the fixed basis blocks we have for $R \in \mathbf{R}^n$

$$PR = \sum_{k=1}^{p} b_k B_k = \sum_{k=1}^{p} \langle R, B_k \rangle B_k.$$

Then the range block R has a unique orthogonal decomposition $R = OR + PR$ where the operator $O = I - P$ projects onto the orthogonal complement $\mathcal{B}^\perp$. If $Z = (z_1, \ldots, z_n) \in \mathbf{R}^n \backslash \mathcal{B}$, we define the operator

$$\phi(Z) = \frac{OZ}{||OZ||}. \tag{2.3}$$

Now for a given domain block D the collage block AA^+R can be given explicitly as

$$AA^+R = \langle R, \phi(D) \rangle \phi(D) + \sum_{k=1}^{p} \langle R, B_k \rangle B_k. \tag{2.4}$$

3 Fast encoding

If the whole codebook is searched, the computations involved in (2.2) make the encoding process time expensive. Following the taxonomy presented in the introduction we describe four approaches for the complexity reduction of the encoding.

3.1 Discrete features: Classification

The classification as described below has been carried out only for the case $p = 1$, where just one fixed basis block of constant intensity $B = (1, \ldots, 1)/\sqrt{n}$ is used. However, at this point we already notice that the method extends to the general case allowing $p > 1$, provided that certain modifications are made. Essentially, this amounts to considering the transformed domains $\phi(D_i)$ (see (2.3)) in place of the original domains.

3.1.1 Jacquin's approach

In his original work [1, 2] Jacquin used a classification scheme coming from a study of Ramamurthi and Gersho [17]. The domain blocks are classified according to their perceptual geometric features. Only three major types of blocks are differentiated: shade blocks, edge blocks, and midrange blocks. In shade blocks the image intensity varies only very little, while in edge blocks a strong change of intensity occurs, e.g., along a boundary of an object displayed in the image. The class of edge blocks is further subdivided into two subclasses: simple and mixed edge blocks. Midrange blocks have larger intensity variations than shade blocks, but there is no pronounced gradient as in an edge block. Thus, these blocks typically are blocks containing texture. Since ranges that would be classified as shade blocks can be approximated well by the constant fixed block B, scaled by an appropriate factor b, it is not necessary to search for a corresponding domain (in effect setting the coefficient $a = 0$). Thus, in this scheme there are really only two (major) classes, one of which must be searched for each non-shade block range.

3.1.2 Classification by intensity and variance

A more elaborate classification technique was proposed by Boss, Fisher and Jacobs [6, 7]. It works as follows. A square range or domain is subdivided into its four quadrants (upper left, upper right, lower left, and lower right). In the quadrants the average pixel intensities A_i and the corresponding variances V_i are computed ($i = 1, \ldots, 4$). It is easy to see that one can always orient (rotate and flip) the range or domain such that the average intensities are ordered in one of the three ways:

$$\begin{aligned} &\text{Major class 1: } A_1 \geq A_2 \geq A_3 \geq A_4, \\ &\text{Major class 2: } A_1 \geq A_2 \geq A_4 \geq A_3, \\ &\text{Major class 3: } A_1 \geq A_4 \geq A_2 \geq A_3. \end{aligned}$$

Once the orientation of the range or domain has been fixed accordingly, there are 24 different possible orderings of the variances which define 24 subclasses for each major class. If the scale factor a in the approximation $aD + bB$ of the range block R is negative then the orderings in the classes must be modified accordingly. Thus, for a given range two subclasses out of 72 need to be searched in order to accommodate positive and negative scale factors.

3.1.3 Archetype classification

A method that defines the classes a priori by some empirical studies carried out on a collection of training images is the archetype classification presented by Boss and Jacobs in [8]. An archetype for a set of codebook blocks is given by that particular codebook block that can best cover all others in the usual least squares sense. For a set of blocks D_i this is the block D_k,

$$D_k = \arg\min_{D_k} \sum_{i \neq k} \min_{a,b} ||D_i - (aD_k + bB)||.$$

Starting out from an arbitrary classification (e.g., the one given by Fisher et al above) of subimage blocks taken from a set of training images one can compute the archetype for each class. Then the blocks are reclassified according to the archetype by which they can be covered best. This yields a new classification, and the process of archetype computation and reclassification is repeated until self-consistency, i.e., until no further change occurs in an iteration. The final set of archetypes becomes a part of the encoder. Given an image to be compressed, the encoder defines the domain pool and classifies all codebook blocks, i.e., for each codebook block the archetype is found that can best cover the block under consideration. In this way it can be expected that a given range can be covered very well by a block in the corresponding class. In fact, this is verified in the experiments reported in the paper. Thus, in order to arrive at a certain image fidelity, one needs to search fewer classes, which saves some computing time. On the other hand, the classification process is more elaborate. As a result, a conventional classification scheme is overall faster for low quality image encoding, while the best image fidelity can be attained much faster using the archetype classification.

3.2 Discrete features: Adaptive clustering

The clustering algorithm proposed by Lepsøy and Øien [9, 10] consists of two phases:

1. The initialization: The range blocks and the projected codebook blocks $\phi(D_k)$ are subdivided into disjoint subsets (clusters) characterized by representative blocks (cluster centers). In contrast to classification methods this procedure is image dependent.

2. The encoding: For each range block the search is restricted to the codebook cluster having the same cluster center.

While initializing the cluster structure the task is twofold: minimizing both the collage error and the complexity of the encoder. If the number of clusters is fixed at M and the clusters have nearly the same size, then the search time is reduced by a factor equal to about $1/M$. So for M fixed, the effort is devoted to the first task. The algorithm proceeds as follows:

Algorithm 1. (Clustering algorithm)
Let an initial set of M cluster centers be given.

1. *Each range block is compared to the cluster centers and put into the cluster whose center most resembles the range block.*
2. **If** *some predefined objective function has increased by at least a given threshold since the last iteration*
   then *modify the cluster centers and go to* 1
   else *group the projected codebook blocks into clusters using the centers obtained.*

Comparing blocks is done with a similarity measure introduced by the following theorem:

Theorem 2. (The similarity measure) [9].
Let R be a range block and $D_1, D_2, \ldots, D_{N_D} \in \mathbf{R}^n \backslash \mathcal{B}$ a collection of codebook blocks. Then the codebook block that minimizes the least squares error $||R - (aD_k + \sum_{i=1}^{p} b_i B_i)||$ is the one that maximizes $\langle R, \phi(D_k)\rangle^2$.

As $\langle R, \phi(D_k)\rangle^2 = ||R||^2 \cos^2 \angle(R, \phi(D_k))$, the theorem states that the best candidate for a range block R is the codebook block D_k such that $\phi(D_k)$ is most nearly parallel to R. It follows that if the range block and the projected domain block are nearly parallel to the cluster center then they are nearly parallel to each other. This motivates restricting the search to the blocks of the codebook cluster that corresponds to the given range block.

The objective function in step 2 of the algorithm aims at evaluating the efficiency of the collage. As a result of Theorem 2 the collage error can be minimized by maximizing the sum

$$\sum_{m=1}^{M} \sum_{i=1}^{I_m} \langle R_{m,i}, \phi(D_{R_{m,i}})\rangle^2 \tag{3.1}$$

where M is the number of clusters, I_m the number of range blocks in the cluster with index m and $D_{R_{m,i}}$ the codebook block chosen to encode $R_{m,i}$. However, expression (3.1), which depends on the cluster centers, cannot be used as an objective function in practice since it is too expensive to compute. It is thus shown that under certain assumptions maximizing the function

$$F(C_1, \ldots, C_M) = \sum_{m=1}^{M} \sum_{i=1}^{I_m} \langle OR_{m,i}, C_m\rangle^2 \tag{3.2}$$

for $||C_m|| = 1$ is a good alternative to the initial problem. Here the vectors C_m are the cluster centers. Now if H_m is the linear operator $H_m : \mathbf{R}^n \to \mathbf{R}^{I_m}$ defined by $H_m C = (\langle OR_1, C\rangle, \ldots, \langle OR_{I_m}, C\rangle)$, then the right term in (3.2) can be written as $\sum_{m=1}^{M} ||H_m C_m||_2^2$. As maximizing l_2 norms still involves too much computation it is instead suggested to maximize the l_1 norms in $\sum_{m=1}^{M} ||H_m C_m||_1$ under the same constraint as before. But considering that

$$\max_{||C||_2=1} ||HC||_1 = \max_{||C||_2=1} \{\langle S, HC\rangle : S = (\pm 1, \ldots, \pm 1)\},$$

constructing M vectors C_m of l_2 norm 1 such that $\langle S, H_m C_m\rangle$ is maximized will be an efficient way to maximize the sum $\sum_{m=1}^{M} ||H_m C_m||_1$. This is done through an iterative algorithm which when applied to the M range clusters at step 2 in the main algorithm, attempts to return M "better" cluster centers.

Algorithm 3. (Cluster center update)
Let C_0 be a given cluster center and $k = 1$.

1. *Compute* $S_k = \mathbf{sgn}(HC_{k-1})$.
2. *Compute* $C_k = \dfrac{H^T S_k}{||H^T S_k||_2}$.
3. *Increment k by 1 and repeat all steps until $S_k = S_{k-1}$.*

Here $\mathbf{sgn}(e_1, \ldots, e_n) = (\mathrm{sgn}(e_1), \ldots, \mathrm{sgn}(e_n))$ and $\mathrm{sgn}(e) = 1$ if $e \geq 0$ and -1 otherwise. The sequence $\{\langle S_n, HC_n\rangle\}$ is non-decreasing and bounded from above by $\max_{||C||_2=1} ||HC||_1$, hence convergent. It is reported that this algorithm stops after less than 7 iterations.

3.3 Continuous features: Functional method

In [11], Bedford, Dekking and Keane proposed a criterion which tells when a codebook block cannot provide a good approximation to a range block. The idea is to compare not ranges agains domains, but rather to compare ranges and domains independently against a certain unit vector. Only when these comparisons come out about the same can a range be covered by a given domain. One can thus reduce the encoding time by eliminating a large number of codebook blocks. We do not give their original result but generalize to arbitrary unit vectors and also to the case of fractal image compression with several fixed basis blocks.

Theorem 4. (A necessary condition)
Let $\delta > 0$, and let U be a unit vector in $\mathbf{R}^n$. Let R and D be in $\mathbf{R}^n \backslash \mathcal{B}$ with $\langle R, \phi(R) \rangle \geq \delta$. If $E = \min_{a,b_1,\ldots,b_p \in \mathbf{R}} ||R - (aD + \sum_1^p b_k B_k)|| \leq \delta$ then

$$\Big| |\langle \phi(R), U \rangle| - |\langle \phi(D), U \rangle| \Big| \leq \sqrt{2 - 2\sqrt{1 - \frac{\delta^2}{\langle R, \phi(R) \rangle^2}}}. \tag{3.3}$$

Proof. We compute using the Cauchy-Schwarz inequality

$$\begin{aligned} (|\langle \phi(R), U \rangle| - |\langle \phi(D), U \rangle|)^2 &\leq (\langle \phi(R), U \rangle - \langle \phi(D), U \rangle)^2 \\ &= |\langle \phi(R) - \phi(D), U \rangle|^2 \leq ||\phi(R) - \phi(D)||^2 \cdot ||U||^2 \\ &= ||\phi(R) - \phi(D)||^2 = 2 - 2\langle \phi(R), \phi(D) \rangle \end{aligned} \tag{3.4}$$

If ρ denotes $\langle \phi(R), \phi(D) \rangle$, then the square of the least squares error is $E^2 = \langle R, \phi(R) \rangle^2 (1 - \rho^2)$ (see the proof of Theorem 7). Thus, it follows from the assumption $E \leq \delta$ that

$$\langle R, \phi(R) \rangle^2 (1 - \rho^2) \leq \delta^2.$$

We may assume $\rho \geq 0$ (otherwise replace D by $-D$) and obtain

$$\rho \geq \sqrt{1 - \frac{\delta^2}{\langle R, \phi(R) \rangle^2}}.$$

Inserting this result in the inequality (3.4) completes the proof.

It is easy to check that in the case $p = 1$ where $\mathcal{B}$ is spanned by the fixed block of constant intensity, $B = (1, 1, \ldots, 1)/\sqrt{n}$, we have for any block $Z \in \mathbf{R}^n \backslash \mathcal{B}$

$$\phi(Z) = \frac{1}{\sqrt{V(Z)}} (z_1 - \overline{z}, \ldots, z_n - \overline{z}),$$

where $\overline{z} = (z_1 + \cdots + z_n)/n$ is the average intensity and $V(Z) = \sum_{k=1}^n (z_k - \overline{z})^2$ the n-fold variance. This is the special case given in [11] where the condition $\langle R, \phi(R) \rangle \geq \delta$ was stated as $V(R) \geq \delta^2$. The algorithm to encode a range block R can be described as follows:

Algorithm 5. (A functional algorithm)

1. *Choose a tolerance δ and a unit vector U.*
2. *(Preprocessing) For every codebook block D compute $|\langle \phi(D), U \rangle|$.*

For each range R do:

3. *Compute* $\langle R, \phi(R)\rangle$ *and the upper bound in (3.3).*
4. *If* $\langle R, \phi(R)\rangle < \delta$, *then no search is needed since* $a = 0$ *gives the least squares error* $E = \langle R, \phi(R)\rangle \le \delta$ *for any codebook block* D.
5. *If* $\langle R, \phi(R)\rangle \ge \delta$ *then compute* $|\langle \phi(R), U\rangle|$ *and reject all the codebook blocks* D *for which the inequality (3.3) of Theorem 4 is not fulfilled.*

It is possible to enhance this functional method by considering several unit vectors U for the criterion in the theorem. In this way one can expect to discard a larger set of domain blocks for a given range block.

In an efficient implementation of the functional method above one would not scan the entire domain pool to extract those domains that pass the test of the theorem. Instead, with any functional method it is better to proceed along the following algorithm:

Algorithm 6. (General functional method)
Assume that a function $F : \mathbf{R}^n \to \mathbf{R}$ *is given such that* $|F(R) - F(D)| \le \epsilon_R$ *implies that the range* R *can be covered well by the domain* D. *Let the domain pool be denoted by* $\{D_1, \ldots, D_{N_D}\}$. *In a preprocessing step do:*

1. *For every domain* $D \in \{D_1, \ldots, D_{N_D}\}$ *compute* $F(D)$.
2. *Sort all domains according to the functional value in a linear array and relabel domains such that* $F(D_1) \le F(D_2) \le \cdots \le F(D_{N_D})$.

For each range R *do:*

3. *Compute* $F(R)$ *and the upper bound* ϵ_R *(e.g., the right term in (3.3) of Theorem 4).*
4. *Using, e.g., the bisection method find the indices* k_0, k_1 *such that* $|F(R) - F(D_k)| \le \epsilon_R$ *if and only if* $k_0 \le k \le k_1$.
5. *Check all domains* D_k *with* $k_0 \le k \le k_1$.

In this procedure a list of candidate domains D_k is produced in $O(\log N_D)$ time while the full scan rejecting the domains that do not pass the test takes $O(N_D)$ time.

3.4 Feature vectors

In the feature vector approach introduced by Saupe in [12, 13, 14] a small set of d real-valued keys is devised for each domain which make up a d-dimensional feature vector. These keys are carefully constructed such that searching in the domain pool can be restricted to the *nearest neighbors* of a query point, i.e., the feature vector of the current range. Thus, we may substitute the sequential search in the domain pool by multi-dimensional nearest neighbor searching.

We consider a set of N_D codebook blocks $D_1, \ldots, D_{N_D} \in \mathbf{R}^n$ and a range block $R \in \mathbf{R}^n$. We let $E(D_i, R)$ denote the smallest possible least squares error of an approximation of the range data R by an affine transformation of the domain data D_i. In terms of a formula, this is

$$E(D_i, R) = \min_{a, b_1, \ldots, b_p \in \mathbf{R}} ||R - (aD_i + \sum_{k=1}^{p} b_k B_k)||.$$

The following theorem provides the mathematical foundation for our feature vector approach. Similarly to Theorem 4 we generalize the original version to the case of several fixed basis blocks.

Theorem 7. [12, 13].
Let $n \geq 2$ and $X = \mathbf{R}^n \backslash \mathcal{B}$. Define the function $\Delta : X \times X \to [0, \sqrt{2}]$ by

$$\Delta(D, R) = \min\left(||\phi(R) + \phi(D)||, ||\phi(R) - \phi(D)||\right).$$

For $D, R \in X$ the least squares error $E(D, R)$ is given by

$$E(D, R) = \langle R, \phi(R) \rangle \, g(\Delta(D, R))$$

where

$$g(\Delta) = \Delta \sqrt{1 - \frac{\Delta^2}{4}}.$$

Proof. The least squares approximation of a range block R was given by equation (2.4) in section 2:

$$\langle R, \phi(D) \rangle \, \phi(D) + \sum_{k=1}^{p} \langle R, B_k \rangle B_k. \tag{3.5}$$

By orthogonality we can express the range block as

$$R = \langle R, \phi(R) \rangle \, \phi(R) + \sum_{k=1}^{p} \langle R, B_k \rangle B_k. \tag{3.6}$$

Using this in the first expression (3.5) we obtain

$$\langle R, \phi(R) \rangle \, \langle \phi(D), \phi(R) \rangle \, \phi(D) + \sum_{k=1}^{p} \langle R, B_k \rangle B_k \tag{3.7}$$

for the least squares approximation of R. The square of the difference of (3.6) and (3.7) gives us the least squares error and is calcuated as

$$E(D, R) = \langle R, \phi(R) \rangle \sqrt{1 - \langle \phi(D), \phi(R) \rangle^2}.$$

Since

$$||\phi(R) \pm \phi(D)|| = \sqrt{2(1 \pm \langle \phi(D), \phi(R) \rangle)}$$

we have

$$\Delta(D, R) = \sqrt{2(1 - |\langle \phi(D), \phi(R) \rangle|)}.$$

Solving for $|\langle \phi(D), \phi(R) \rangle|$ and inserting the square of the result in the formula for $E(D, R)$ completes the proof.

The theorem states that the least squares error $E(D_i, R)$ is proportional to the simple function g of the Euclidean distance Δ between the projections $\phi(D_i)$ and $\phi(R)$ (or $-\phi(D_i)$ and $\phi(R)$). Since $g(\Delta)$ is a monotonically increasing function for $0 \le \Delta \le \sqrt{2}$ we conclude that *the minimization of the least squares errors $E(D_i, R)$ for $i = 1, \ldots, N_D$ is equivalent to the minimization of the distance expressions $\Delta(D_i, R)$*. Thus, we may replace the computation and minimization of N_D least squares errors $E(D_i, R)$ by the search for the nearest neighbor of $\phi(R) \in \mathbf{R}^n$ in the set of $2N_D$ vectors $\pm\phi(D_i) \in \mathbf{R}^n$.

There are well known solutions to the problem of finding closest neighbors in Euclidean spaces. A method using kd-trees that runs in expected logarithmic time is presented in [18] together with pseudo code. After a preprocessing step to set up the required kd-tree, which takes $O(N_D \log N_D)$ steps, any search for the nearest neighbors of query points can be completed in expected logarithmic time, $O(\log N_D)$. However, as the dimension n increases, the performance may suffer. A method that is more efficient in that respect, presented in [19], produces a list of so-called approximate nearest neighbors.

We conclude with some remarks on generalizations and implications.

Keeping the n-dimensional keys in cpu memory is not practical for large domain pools. Thus, it is proposed to reduce the dimension of these feature vectors by pixel averaging. For example, all square subimages can be downfiltered to size $d = 4 \times 4$. Then there are only $d = 16$ real keys for each codebook block. Moreover, these keys can be quantized without harm. We have used a representation of one byte per key. An alternative reduction of dimension has been suggested and implemented by Barthel et al [20]. They have used a two-dimensional discrete cosine transformation (DCT) of the projected codebook blocks $\pm\phi(D_i)$. From the resulting coefficients, only the first d of them make up the feature vector. Properties of the DCT transform carry over the result of Theorem 7 to the frequency domain and nearest neighbors of DCT coefficient vectors will yield the smallest least squares errors.

Because of the downfiltering and the quantization, it can happen that the nearest neighbor in feature vector space is not the codebook block with the minimum least squares error. Moreover, it could yield a scaling factor a being too large to be allowed. To take that into consideration, we search the *entire* domain pool not only for the nearest neighbor of the given query point but also for, say, the next 6 or 10 nearest neighbors (this can still be solved in logarithmic time using a priority queue). From this set of neighbors the non-admissible domains are discarded and the remaining domains are compared using the ordinary least squares approach.

We make two technical remarks concerning *memory requirements* for the kd-tree. Firstly, it is not necessary to create the tree for the full set of $2N_D$ keys in the domain pool. We need to keep only one multi-dimensional key per domain, e.g., by keeping only the key which has a non-negative first component (multiply key by -1 if necessary). In this set-up a kd-tree of all $2N_D$ vectors has two symmetric main branches (separated by a coordinate hyperplane), thus, it suffices to store only one of them. Secondly, there is some freedom in the choice of the *geometric transformation* that maps a domain onto a range coming from the 8 possible rotations and reflections of a square subimage. This will create a total of 8 entries per domain in the kd-tree, enlarging the size of the tree. However, we can get away without this tree expansion. To see this, just note that we may instead consider the 8 transformations of the *range* and search the original tree for nearest neighbors of each one of them.

Solving the equality for Δ in the expression for the error $E(D,R)$ in Theorem 7 yields a necessary and sufficient condition in the spirit of Theorem 4.

Corollary 8. (A necessary and sufficient condition)
Let $\delta > 0$ and $n \geq 2$. Let R and D be in $\mathbf{R}^n \backslash \mathcal{B}$ with $\langle R, \phi(R)\rangle \geq \delta$. Then $E(D,R) = \min_{a,b_1,\ldots,b_p \in \mathbf{R}} \|R - (aD + \sum_{k=1}^{p} b_k B_k)\| \leq \delta$ if and only if

$$\Delta(D,R) \leq \sqrt{2 - 2\sqrt{1 - \frac{\delta^2}{\langle R, \phi(R)\rangle^2}}}, \tag{3.8}$$

where $\Delta(D,R)$ is defined as in Theorem 7.

Proof. From Theorem 7, $E(D,R) = \langle R, \phi(R)\rangle\, g(\Delta(D,R))$ with $g(\Delta) = \Delta\sqrt{1 - \frac{\Delta^2}{4}}$. Thus, for $0 \leq \Delta \leq \sqrt{2}$ we have $E(D,R) \leq \delta$ if and only if $\Delta^4 - 4\Delta^2 + 4\delta^2/\langle R, \phi(R)\rangle^2 \geq 0$. From this the assertion easily follows.

In fact, note that the right-hand side of equation (3.8) is identical to that of equation (3.3) in Theorem 4. Now, since the left-hand side in (3.3) is bounded by Δ, Theorem 4 follows immediately from this Corollary.

Remark. In [21], Novak assigns a 4-dimensional feature vector to each block. The components of the feature vector are some invariants defined from the moments of the grey level distribution within the block. Another approach using two features is given by Frigaard et al in [22]. They consider the grey level standard deviation (a continuous variable) and the number of dominant grey levels (a discrete feature) in a block. Both methods are intuitive in the sense that no supporting theory is given to the fact that closeness in the feature space ensures good approximations in the l_2 norm.

4 Comparison of the methods

Our evaluation of the methods is based on computer experiments we have carried out with all of the algorithms presented in the last section except for the adaptive clustering approach of Lepsøy and Øien, which is the most complicated to implement. Instead, in the next section we propose an alternative adaptive clustering method which is more straightforward and which we believe to yield comparable or even better results.

To test the classification scheme of Fisher et al we used his adaptive quadtree program which is in the public domain. We implemented the nearest neighbor method using feature vectors as well as the functional method of Bedford et al within the framework of Fisher's program. The results of the comparison between the classification and the nearest neighbor method and implementation details are reported in depth in the paper [14]. Here we only review the key results: When allowing the same amount of computing time for either method, we have found that with the nearest neighbor approach the image quality improves slightly for images of resolution 512 by 512 and more noticable by 0.8 dB peak-signal-to-noise ratio (PSNR) for larger images of 1024 by 1024 resolution. The compression ratio improved likewise. On another matter it can be observed that an enlargement of the domain pool by a factor of 4 will lead to quadrupled computing time for the classification method (this reflects the linear time complexity) but will rise only by a factor of 1.5 with the nearest neighbor method based on feature vectors.

We continue with the empirical results obtained for the functional method of Bedford et al which we have implemented as described in [23]. In order to get an idea of the performance, we set up the fractal image compression routine so that it would use ranges of only one size, e.g., 64 $\times$ 64 pixels, i.e., we did not apply the adaptive quadtree option in Fisher's program. For each range block size we ran the program four times with differing values of the threshold δ. In the following table we record two numbers for each of the tests: first the percentage of ranges R that qualify as "flat", i.e., where $V(R) < \delta^2$, so that no search is required. For all remaining ranges a search through the codebook becomes necessary. The second number then is the percentage of domains that drop out due to the criterion given in Theorem 4.

The table shows several remarkable results. Firstly, the preliminary flatness criterion identifies a considerable number of ranges for which no search

Table 1.

range size	n	$\delta = 2\sqrt{n}$ %1	%2	$\delta = 5\sqrt{n}$ %1	%2	$\delta = 8\sqrt{n}$ %1	%2	$\delta = 10\sqrt{n}$ %1	%2
64×64	4096	0.0	79.6	0.0	55.8	0.0	39.8	0.0	31.2
32×32	1024	0.0	67.9	4.7	46.3	14.1	35.5	18.0	29.7
16×16	256	0.1	55.1	16.5	35.7	33.1	28.1	42.9	25.1
8×8	64	0.6	40.6	34.8	27.6	57.0	25.1	64.8	22.8
4×4	16	3.6	27.8	55.5	21.6	75.6	22.3	81.0	20.9

is necessary, especially for small block sizes or large tolerances. Secondly, the number of codebook blocks rejected due to the functional criterion typically varies between 20% and 40%. Thus, a speed up factor of about 0.6 to 0.8 over the full search of the entire codebook can be expected due to the use of the search complexity reduction. This is by far not as good as the savings given by the classification method. However, to come to a definite conclusion the trade off between computing times and quality of the encoding in terms of compression ratio and image fidelity has to be carefully considered. The classification is faster, but at the expense of a somewhat reduced image quality. Using the functional criterion such a degradation does not occur. Thus, in the latter case perhaps the same (lower) image quality and compression ratio can be obtained when using a smaller domain pool. This point remains to be fully investigated.

Although we have not implemented the clustering algorithm proposed by Lepsøy and Øien, we would like to point out some problems that may hinder its efficiency. The results reported in the papers show that the loss of image quality is not critical. Hence, the assumptions made while maximizing the objective function seem to be tolerable (see the original papers [9, 10] for more details on these assumptions). As remarked by the authors, the resulting clustering is sensitive to the choice of the initial cluster centers. It is therefore suggested to use a predesigned set of initial cluster centers in order to improve the performance of the algorithm. In the algorithm the number M of clusters is fixed. The value of M that minimizes the total complexity of the algorithm cannot be known a priori by the encoder. We conclude the remarks by noting the difficulty to adapt the algorithm into a quadtree like scheme.

Finishing the discussion we want to emphasize that the discrete methods (classification and adaptive clustering) should not be considered as competitors to the continuous ones (functional method and feature vectors). On the contrary, each one of the first group can be combined with either one of the second group. This can lead to a considerable synergetic effect. In practice one first reduces the search by considering only a class or a cluster corresponding to a given range. In a preprocessing step each class (or cluster) should have been prepared individually for application of the functional method or the nearest neighbor search. Then this second method further speeds up the search in the class or the cluster.

We have only tested the combination of classification and feature vectors in this spirit with very good results (see [14]). This approach makes the algorithm run 2 to 4 times as fast as the already very fast pure classification method. The speed up depends on the image size, and the gain increases for larger images. The other three combinations of methods are perhaps equally promising.[2]

5 Adaptive clustering and Hilbert curve ordering in feature vector space

The adaptive clustering of Lepsøy and Øien was based on the similarity measure. At that time (1993) the relation of this criterion to Euclidean distances in feature vector space (Theorem 7) had not been known. With the new result at hand it is possible to create new clustering algorithms geared to Eucliden distances in d-dimensional real space. We show how applying the multidimensional clustering algorithm of Wan et al [25] is feasible in the search problem. This algorithm falls into the category of the heuristic methods as opposed to the iterative optimization ones. The first methods were designed to reduce the computational complexity while providing an acceptable solution. The input data points are the $2N$ feature vectors $\pm\phi(D_i) \in \mathbf{R}^n$ corresponding to the codebook blocks. To simplify the notations we denote these vectors by $G_1, \ldots, G_{2N}$. The objective is to find M cluster centers $C_1, \ldots, C_M$ such that the average sum-of-squared-errors defined by

$$E = \frac{1}{2N} \sum_{i=1}^{2N} ||G_i - C(G_i)||^2 \tag{5.1}$$

is minimum, where $C(G_i)$ is the cluster center closest to G_i. Basically the algorithm partitions the vector space sequentially into M disjoint hyperboxes. The cluster centers are chosen to be the centroids of these hyperboxes. Once the clustering is finished, the range block R is encoded in two steps. First, we map its feature vector $\phi(R)$ to its closest cluster center $C(R)$, e.g., by using the fast nearest neighbor search in the set $\{C_1, \ldots, C_M\}$. Then the search for the matching feature vector G is restricted to the cluster whose center is $C(R)$. In the implementation of

[2] For completeness we mention the work of Bani-Eqbal in [24] in which an approach different from all the others is considered. In it the domains are organized in a tree structure which is pruned using some mathematical results and empirical rules of thumb.

this algorithm and as in section 3.4 each feature vector can be considered as a 16 dimensional real valued-key where each key is represented by one byte. Other heuristic clustering techniques such as the median-cut and the mean-split algorithms may be used as an adaptive classification scheme in fractal image compression. However, since the partition strategy of the algorithm introduced in [25] is explicitly based on the minimization of the sum of squared errors, we think that it is better suited to producing smaller collage errors.

The nearest neighbor method works in a d-dimensional space of feature vectors. To modify the approach in order to arrive at a functional method using only one coordinate we can utilize a one-dimensional parametrization of this space. This can be accomplished by space-filling curves. These curves exhibit a certain amount of coherence, i.e., points (feature vectors) that are close in d-dimensional space typically correspond to parameters of the curve that are close. Thus, given a feature vector $\phi(R)$ corresponding to a range R, we can search for its neighbors on the space-filling curve. This procedure may yield for the range block R a number of codebook blocks D_k that cover the range well. Of course, it is not guaranteed to obtain the best covering. For a practical example, we can consider a multi-dimensional Hilbert curve, for which we give an inductive definition.

Definition 9. (d-dimensional stage-n Hilbert curve)
Let $I_n = \{0, \ldots, 2^n - 1\}$ and $I_n^d = I_n \times \cdots \times I_n$ (d times). A d-dimensional stage-0 Hilbert curve is the trivial mapping $H_0^d : I_0 \to I_0^d$, i.e., $H_0^d(0) = (0, \ldots, 0)$. A d-dimensional stage-n Hilbert curve *is a mapping $H_n^d : I_{dn} \to I_n^d$ with the following properties:*

1. *The mapping H_n^d is bijective.*
2. *Two consecutive points $H_n^d(k)$ and $H_n^d(k+1)$ (for $k = 0, \ldots, 2^{nd} - 2$) differ in only one coordinate. The difference is ± 1.*
3. *Let $M_n^d : I_n^d \to I_{n-1}^d$ be the bit-masking operator $M_n^d(i_1, \ldots, i_d) = (i_1 \bmod 2^{n-1}, \ldots, i_d \bmod 2^{n-1})$. For each $k = 0, \ldots, 2^d - 1$ the mapping $i \mapsto M_n^d(H_n^d(2^{(n-1)d}k + i))$ is a d-dimensional stage-$(n-1)$ Hilbert curve.*

A constructive algorithm for a d-dimensional stage-n Hilbert curve has been given by Butz [26]. It can been seen that this algorithm is invertible, a property that is required for our application.

In the context of our application we propose to use the 16-dimensional feature vectors as in section 3.4, thus, $d = 16$. With the 8-bit quantization of the keys in this space we get $n = 8$. In practice one has to sort the quantized feature vectors $\pm\phi(D_k)$ according to increasing parameter values given by the Hilbert scan of the 16-dimensional 8-bit resolution space, and then to apply some version of Algorithm 6. It is possible to refine this approach. For example, as already mentioned in the last section we may combine the functional method with, e.g., classification. Or, we can consider two space-filling curves providing up to four neighbors for a given range. Each one of these neighbors again has up to four neighbors, etc. This set of neighbors will yield an improved set of candidate domains at the cost of increased preprocessing.[3]

6 Conclusion

In this essay we have presented a survey for complexity reduction schemes in fractal image compression and classified the approaches using a new taxonomy. The results have been extended and two new techniques have been proposed: adaptive clustering and Hilbert curve ordering based on a feature vector space. In particular this paper shows how different complexity reduction techniques can be combined to an advantage.

Acknowledgement. We appreciate the invaluable contribution of Matthias Ruhl, who organized the computer programs and ran the experiments.

Bibliography

1. Jacquin, A. E., A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding, PhD Thesis, Georgia Institute of Technology, August 1989.

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

3. Barnsley, M., Hurd, L., *Fractal Image Compression*, AK Peters, Wellesley, 1992.

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

5. Saupe, D., Hamzaoui, R., A review of the fractal image compression literature, *ACM Computer Graphics 28*, 4, 268–279 (Nov. 1994). Technical Report 58, Institut für Informatik, Universität Freiburg, 1994.

6. Jacobs, E. W., Fisher, Y., Boss, R. D., Image compression: A study of the iterated transform method, *Signal Processing 29* (1992) 251–263.

[3] Work on implementations of both approaches proposed in this section is currently being done, the results of which are to appear elsewhere.

7. Fisher, Y., Fractal image compression with quadtrees, in: Y. Fisher, *Fractal Image Compression — Theory and Application,* Springer-Verlag, New York, 1994.

8. Boss, R. D., Jacobs, E. W., Archetype classification in an iterated transformation image compression algorithm, in: *Fractal Image Compression — Theory and Application,* Y. Fisher (ed.), Springer-Verlag, New York, 1994.

9. Lepsøy, S., Attractor Image Compression: Fast Algorithms and Comparisons to Related Techniques, PhD Thesis, The Norwegian Institute of Technology, Trondheim, Norway, June 1993.

10. Lepsøy, S., Øien, G. E., Fast attractor image encoding by adaptive codebook clustering, in: *Fractal Image Compression — Theory and Application,* Y. Fisher (ed.), Springer-Verlag, New York, 1994.

11. Bedford, T., Dekking, F. M., Keane, M. S., Fractal image coding techniques and contraction operators, *Nieuw Arch. Wisk. (4)* 10,3 (1992) 185–218.

12. Saupe, D., Breaking the time complexity of fractal image compression, Technical Report 53, Institut für Informatik, Universität Freiburg, 1994.

13. Saupe, D., From classification to multi-dimensional keys, in: *Fractal Image Compression — Theory and Application,* Y. Fisher (ed.), Springer-Verlag, New York, 1994.

14. Saupe, D., Accelerating fractal image compression by multi-dimensional nearest neighbor search, in: *Proceedings Data Compression Conference, March 28–30, 1995 • Snowbird, Utah,* J. A. Storer, M. Cohn (eds.), IEEE Computer Society Press, 1995.

15. Monro, D. M., Dudbridge, F., Fractal approximation of image blocks, *Proceedings of ICASSP-1992 IEEE International Conference on Acoustics, Speech and Signal Processing, 3*, 1992.

16. Monro, D. M., Woolley, S. J., Fractal image compression without searching, *Proceedings of ICASSP-1994 IEEE International Conference on Acoustics, Speech and Signal Processing, 5*, Adelaide, 1994.

17. Ramamurthi, B., Gersho, A., Classified vector quantization of images, *IEEE Trans. Commun.*, COM-34, (1986) 1105–1115.

18. Friedman, J. H., Bentley, J. L., Finkel, R. A., An algorithm for finding best matches in logarithmic expected time, *ACM Trans. Math. Software 3*, (1977) 209–226.

19. Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., Wu, A., An optimal algorithm for approximate nearest neighbor searching, *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms* (1994) 573–582.

20. Barthel, K. U., Schüttemeyer, J., Voye, T., Noll, P., A new image coding technique unifying fractal and transform coding, in: *IEEE International Conference on Image Processing ICIP'94,* Austin, Texas, Nov. 1994.

21. Novak, M., Attractor coding of images, Licentiate Dissertation, Dept. of Electrical Engineering, Linköping University, May 1993.

22. Frigaard, C., Gade, J., Hemmingsen, T., Sand, T., Image compression based on fractal theory, Manuscript, Institute for Electronic Systems, Aalborg University, Denmark, 1994.

23. Bedford, T., Dekking, F. M., Breewer, M., Keane, M. S., van Schooneveld, D., Fractal coding of monochrome images, *Signal Processing, 4*, (1994) 405–419.

24. Bani-Eqbal, B., Speeding up fractal image compression, *Proceedings from IS&T/SPIE 1995 Symposium on Electronic Imaging: Science & Technology,* Vol. 2418: Still-Image Compression 1995.

25. Wan, S. J., Wong, S. K. M., Prusinkiewicz, P., An algorithm for multidimensional data clustering, *ACM Transactions on Mathematical Software 14*, 2 (1988) 153–162.

26. Butz, A. R., Alternative algorithm for Hilbert's space-filling curve, *IEEE Trans. on Computers*, April 1971, 424–426.