Lossless acceleration of fractal image encoding via the Fast Fourier Transform

Hannes Hartenstein[a], Dietmar Saupe[b]

[a]*Institut für Informatik, Universität Freiburg, 79110 Freiburg, Germany*

[b]*Institut für Informatik, Universität Leipzig, 04109 Leipzig, Germany*

Abstract

In fractal image compression the encoding step is computationally expensive. We present a new technique for reducing the encoding complexity. It is lossless, i.e., it does not sacrifice any image reconstruction quality for the sake of speedup. It is based on a codebook coherence characteristic of fractal image compression and leads to a novel application of the Fast Fourier Transform based cross-correlation. The proposed method is particularly well suited for use with highly irregular image partitions for which most traditional (lossy) acceleration schemes lose a large part of their efficiency. For large ranges our approach outperforms other currently known lossless acceleration methods.

1 Introduction

Fractal image compression [1,8,11] exploits redundancy given by self-similarities within an image. The image to be coded is partitioned into a set of image blocks (called *ranges* in fractal coding parlance). For each range one searches for another part of the image called a *domain* that gives a good approximation to the range when appropriately scaled in order to match the size of the range and transformed by a luminance transformation that provides for contrast and brightness adjustment (cf. Figure 1). The list of parameters specifying for each range the corresponding domain and the affine luminance transformation together with the partition information is called a *fractal code*. Such a fractal code defines an operator working on the space of images. When this operator is applied to an arbitrary image, it partitions the image and replaces each range by the (transformed) domain. When determining a fractal code one also has to ensure that the corresponding operator represents a contraction mapping. Thus, when the decoder iteratively applies the operator to an arbitrary image, the resulting sequence of images converges to the operator's unique fixed point which represents an approximation to the original image as decoder output.

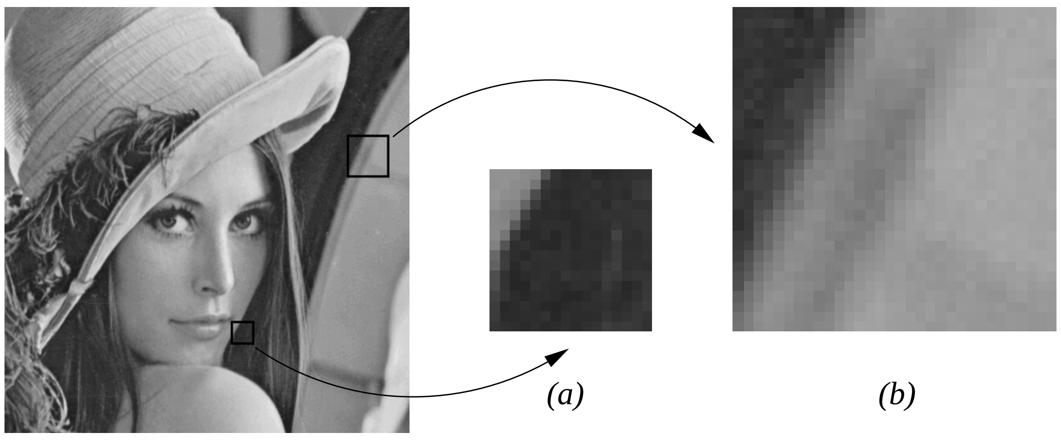


Fig. 1. Two parts of an image that are *similar*: block (a) can be described using block (b). Basically, the whole block (b) has to be shrunk to match the size of (a), and the luminance values have to be inverted.

For details of fractal modeling of images as well as decoding issues we refer the reader to [8]; in this paper we will focus on the encoding step in fractal coding schemes.

The fractal encoding step, i.e., the task to determine for each range a good fitting domain, is similar to the codebook search in vector quantization. However, in contrast to vector quantization the codebooks are not designed using a training sequence but are directly derived from the image to be coded. Typically, the codebook for a range is constructed using the image downscaled to half its original resolution. Each block in the downscaled image of the same size and shape as the range is considered a codebook block. Note that the codebook blocks are *overlapping* and correspond to image parts of twice the linear size of the range. We refer to codebooks constructed in the above manner as *canonical* codebooks. Also note that different codebooks are applicable for ranges of different size or shape.

The goal in the encoding step is to find for each range a codebook block that gives the least L_2-error when adjusted by an affine luminance transformation. [1] Thus, a computationally expensive least-squares optimization is required for each pair of range and codebook block in order to determine the optimal luminance transformation and the resulting approximation error. Since codebooks consist of many thousands of blocks, the straightforward implementation of fractal image compression by "brute force" [1] suffers from long encoding times. Therefore, much effort has been undertaken to find a variety of ways to speed up the process; for a survey see [22]. Most techniques are *lossy* acceleration schemes in the sense that they sacrifice image reconstruction quality for the sake of speedup. For example, an acceleration method that only

[1] We restrict ourselves to conventional fractal coding based on the collage theorem [1,8]; we do not consider direct attractor optimization here.

considers a subset of the canonical codebooks results in a speedup by choosing an acceptable but suboptimal codebook block and, therefore, loses some of the possible coding quality since the canonical codebooks are superior to smaller codebooks with respect to rate vs. distortion performance. The majority of acceleration methods reported to date, such as classification [9], clustering [10], or tree-structured search [4,20], follows this principle of considering only a small subset of the canonical codebook for a given range. In contrast, using *lossless* acceleration methods one does not trade coding quality for speedup. With lossless acceleration techniques one is guaranteed that for each range the entire corresponding canonical codebook is considered and the optimal codebook block is determined giving the least approximation error using *quantized* luminance parameters. Thus, efficient lossless acceleration methods allow to explore the rate vs. distortion performance limits of fractal coding schemes; in addition they allow an efficient assessment of the loss in rate vs. distortion performance for the case when lossy acceleration schemes are used. In general, lossless acceleration techniques provide a means to speed up the production of *good* fractal codes and, in addition, are useful in fractal coding research when one compares design choices such as partitioning schemes, quantization, and rate vs. distortion optimization.

In this paper we put forward a lossless acceleration method that is based on the codebook coherence of canonical codebooks and leads to a novel application of the Fast Fourier Transform (FFT) based cross correlation. The FFT is used to speed up certain calculations. Nevertheless, the proposed encoding method still represents a spatial domain approach; we do not transfer the least squares optimizations to the frequency or wavelet domain, as it has been done before, for instance, in [2,6,12]. For large ranges our approach outperforms other lossless acceleration methods currently known. Our lossless acceleration method has the additional advantage that it can easily be used with highly image-adaptive and irregular partitions which have been shown to lead to excellent fractal coding results [18]. In the presence of irregular partitions, traditional (lossy) acceleration techniques typically suffer from a performance drop since most of them are based on the assumption that ranges have only a few sizes and shapes, e.g., square blocks of size 4×4, 8×8, and 16×16 pixels. In such a case certain quantities can be computed and stored in a preprocessing step for later use in the block matching operations. With an irregular partitioning this procedure loses a large part of the efficiency of the acceleration method.

This paper is organized as follows. The notation is introduced below. In Section 3 we discuss methods previously published for lossless acceleration of fractal encoding. In Section 4 we will first analyze the computational costs of the standard fractal encoding approach in order to determine the bottlenecks of the encoding process and then put forward our FFT-based acceleration technique. Implementation details and experimental results are given. Features and extensions, e.g., the application of our method to highly irregular image partitions, are discussed in Section 5.

2 Basics & Notation

There are numerous variants and extensions of fractal image compression, and many of them may take advantage of the acceleration technique presented in this paper. For simplicity and clarity of our exposition we consider a generic type of fractal coding for grey scale images. Let the image to be coded be denoted by $I \in \mathbb{R}^{N \times N}$, where N is a power of 2, and a partitioning of the image I into a set of ranges is considered as already given. Let $h \in \mathbb{R}^{N/2 \times N/2}$ denote the downscaled version of the image I, i.e., $h(i,j) = \frac{1}{4}\sum_{x=2i}^{2i+1}\sum_{y=2j}^{2j+1} I(x,y)$, $0 \le i,j < N/2$. For a given range the corresponding canonical codebook consists of all image blocks in the downscaled version h that have the same size and shape as the range. Note that the range can be of arbitrary shape, e.g., ranges can be polygons. Using an arbitrary but fixed scanning procedure the range and the codebook blocks are converted into vectors which we denote by $R, D^R_{0,0}, \ldots, D^R_{N/2-1,N/2-1}$, and call blocks again, for convenience. The number of codebook blocks in the canonical codebook for a given range is equal to $N^2/4$ since we allow codebook blocks to *wrap around* image borders. For better readability we will simply write D or D_{n_1,n_2} instead of $D^R_{n_1,n_2}$ when the relationship to a range is evident.

A range has to be compared to all codebook blocks in the corresponding canonical codebook in order to determine the best codebook block giving the least approximation error under an affine luminance transformation. The distortion function for a fixed range R and codebook block D is a quadratic function of the parameters $s, o \in \mathbb{R}$ of the affine luminance transformation:

$$\begin{aligned} d_{D,R}(s,o) &= \|R - (sD + o\mathbf{1})\|_2^2 \qquad (1) \\ &= \langle D,D\rangle s^2 + 2\langle D,\mathbf{1}\rangle so + no^2 - 2\langle R,D\rangle s - 2\langle R,\mathbf{1}\rangle o + \langle R,R\rangle . \end{aligned}$$

Here, $\mathbf{1}$ denotes the constant block with unit intensity at every pixel, and $\langle\cdot,\cdot\rangle$ denotes the inner product in a Euclidean space of dimension n (n = number of pixels in the range block). The parameters s, o are called scaling factor and offset, respectively.

A least squares optimization gives the optimal luminance parameters $\hat{s}, \hat{o} \in \mathbb{R}$. Since quantizing the optimal luminance parameters will increase the approximation error, one has to choose the codebook block that minimizes the approximation error using *quantized* luminance parameters $\hat{s}_q, \hat{o}_q$, i.e., the codebook block that minimizes the *collage error*

$$E(D,R) = \|R - (\hat{s}_q D + \hat{o}_q \mathbf{1})\|_2^2. \tag{2}$$

The quantizer for the scaling factor has to clamp the value $\hat{s}$ to $[-s_{max}, +s_{max}]$, $0 < s_{max} < 1$, in order to ensure convergence in the iterative decoding; s_{max} is a fixed parameter of the fractal coding scheme. The fractal code for range R then consists of the index of the best codebook block with respect to the collage error together with the corresponding quantized luminance parameters.

In most fractal image coders isometric transformations of the downscaled image (rotations by multiple of $\pi/2$ and reflections) are considered which enlarge the pool by a factor of 8. However, this option is not an intrinsic part of fractal coding and does not necessarily lead to improved performance in the rate-distortion sense [23]. Thus, for simplicity, we do not consider isometries here and comment later on the increase of costs that the option of isometries introduces in connection with our FFT-based method.

3 Previous work

A lossless acceleration method based on *image pyramids* is presented in [7]. It can be described as follows. Here, let R denote an image block of size $2^m \times 2^m$ pixels. Define $R^{(m)} := R$ and

$$R^{(k)}(i,j) := \frac{1}{4} \sum_{x=2i}^{2i+1} \sum_{y=2j}^{2j+1} R^{(k+1)}(x,y),$$

with $k = 0, \ldots, m-1$, and $i,j = 0, \ldots, 2^k - 1$. The pyramid $D^{(k)}$ based on the codebook block D is defined in the same fashion. Now the following inequality holds:

$$\frac{1}{2^k \cdot 2^k} d_{D^{(k)},R^{(k)}}(\hat{s}^{(k)}, \hat{o}^{(k)}) \leq \frac{1}{2^{k+1} \cdot 2^{k+1}} d_{D^{(k+1)},R^{(k+1)}}(\hat{s}^{(k+1)}, \hat{o}^{(k+1)}),$$

where $(\hat{s}^{(k)}, \hat{o}^{(k)}) = \arg\min_{s,o \in \mathbb{R}} \|R^{(k)} - (sD^{(k)} + o\mathbf{1})\|_2^2$. For the computation of $E(D^{(k)}, R^{(k)})$ fewer floating point operations are required than for the computation of $E(D^{(k+1)}, R^{(k+1)})$ since the number of pixels in the blocks $D^{(k)}, R^{(k)}$ is four times less than in $D^{(k+1)}, R^{(k+1)}$. Therefore, one can first compute the collage error of downfiltered versions of the corresponding codebook block D and range R, and when the resulting error is already larger than the minimal collage error encountered for range R so far, the codebook block can be excluded from further consideration since the actual collage error at full resolution can only be larger than the low resolution estimate. Thus, the number of floating point operations required to detect a codebook block as a suboptimal choice is reduced. The speedup factor compared to the standard scheme is reported to be about 4.

In [3] a criterion is presented that also provides a method for excluding codebook blocks from the search by performing a few comparisons. The range is not directly compared against the codebook blocks, but the range and the codebook blocks are compared against a unit vector, i.e., one compares $|\langle \bar{R}, U\rangle|$ with $|\langle \bar{D}, U\rangle|$, where U is a unit vector. Only when those two values are close to each other, the codebook block is considered for the range. Formally, the criterion can be stated as follows [22]. By $\bar{R}, \bar{D}$ we denote the range, resp. codebook block, normalized to zero mean and unit variance. Let $\delta > 0$ and $U \in \mathbb{R}^n$ with $\|U\| = 1$. Let R and D be elements of $\mathbb{R}^n$ with $\langle R, \bar{R}\rangle \geq \delta$. If $\min_{s,o\in\mathbb{R}} \|R - (sD + o\mathbf{1})\|^2 \leq \delta$ then

$$\Big| |\langle \bar{R}, U\rangle| - |\langle \bar{D}, U\rangle| \Big| \leq \sqrt{2 - 2\sqrt{1 - \frac{\delta^2}{\langle R, \bar{R}\rangle^2}}}. \tag{3}$$

Unfortunately, the speedup factor obtained with this method is rather small (in our tests [22] the speedup factor has been 1.5 over direct computation), and it requires that only a small number of different range sizes are present in the image partition.

With the notation as defined above the distortion for a range R and a codebook block D using unquantized luminance parameters can be written as

$$d_{D,R}(\hat{s}, \hat{o}) = \left(\langle R, R\rangle - \frac{1}{n}\langle R, \mathbf{1}\rangle^2\right) \cdot \left(1 - \langle \bar{D}, \bar{R}\rangle^2\right). \tag{4}$$

For a given range R this expression is minimal when $|\langle \bar{D}, \bar{R}\rangle|$ is maximal. In [4] a fast technique for maximizing $|\langle \bar{D}, \bar{R}\rangle|$ is presented which uses partial distortion elimination. However, equation (4) can only be used for the evaluation of the error using the *unquantized* optimal scaling parameter $\hat{s}$ and offset $\hat{o}$. Therefore, the codebook block D that maximizes $|\langle \bar{D}, \bar{R}\rangle|$ is not optimal when *quantized* luminance parameters are used. We, therefore, do not consider techniques that only maximize $|\langle \bar{D}, \bar{R}\rangle|$ as lossless acceleration schemes.

The lossless acceleration technique based on the Fast Fourier Transform has been put forward by the authors in [19,21]. In this paper we will extend and complete the discussion. More recently, a closely related lossless method was proposed in this journal [17] for which large acceleration factors were reported. According to this method the codebook is first reduced in size and then expanded again by considering all circular shifts of the codebook vectors. However, this non-standard codebook design with circularly shifted vectors leads to a significant drop in the quality of the encodings. Thus, the technique in [17] provides a fast method to search a poor codebook. Our method presented here does not suffer from such a drawback. It provides speedup of the generic fractal coding procedure without sacrificing any losses in quality. The resulting encodings are identical to those obtained by the full search of the canonical codebooks.

4 Fractal encoding via fast cross correlation

As a starting point, we will first present an analysis of the computational costs of conventional encoding. Then we put forward our Fast Fourier Transform-based lossless acceleration method. Finally, we present implementation details that are of crucial importance for the success of our method.

4.1 Motivation

Almost all the work of a fractal image coder goes into the calculation of the many collage error terms. Let us examine the number of floating point operations involved in the calculation of the collage error $E(D, R)$ for a fixed codebook block D and a fixed range R. The optimal luminance parameters $\hat{s}$ and $\hat{o}$ can be obtained via

$$\hat{s} = \begin{cases} \dfrac{n\langle D, R\rangle - \langle D, \mathbf{1}\rangle \langle R, \mathbf{1}\rangle}{n\langle D, D\rangle - \langle D, \mathbf{1}\rangle^2} & \text{if } n\langle D, D\rangle - \langle D, \mathbf{1}\rangle^2 \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\hat{o} = \frac{1}{n} \left(\langle R, \mathbf{1}\rangle - \hat{s}\langle D, \mathbf{1}\rangle \right). \tag{6}$$

As above, n denotes the number of pixels in range R.

Let us assume for now that the inner products $\langle D, R\rangle$, $\langle D, D\rangle$, $\langle D, \mathbf{1}\rangle$, $\langle R, R\rangle$, and $\langle R, \mathbf{1}\rangle$ have already been calculated. Then, the computation of $(\hat{s}, \hat{o})$ using (5), (6) requires 10 floating point operations (flops).

The quantization of the luminance parameters is performed as in the quadtree-based fractal coder of [8]. First, $\hat{s}$ is quantized using a uniform quantizer with $2^{s_{bits}}$ levels for $[-s_{max}, s_{max}]$, i.e.,

$$\hat{s}_q = \lfloor (\hat{s} + s_{max}) \cdot c + 0.5 \rfloor \cdot d - s_{max},$$

with $c = 2^{s_{bits}-1}/s_{max}$ and $d = 1/c$. Thus, the quantization of $\hat{s}$ requires 5 flops. Then the optimal offset for this quantized scaling coefficient is computed. This offset is quantized using a uniform quantizer for the interval $[o_{min}(\hat{s}_q), o_{max}(\hat{s}_q)]$ with $2^{o_{bits}}$ levels, where

$$\begin{aligned}[o_{min}(s), o_{max}(s)] &= [-255s, 255] \text{ for } s \geq 0, \\ [o_{min}(s), o_{max}(s)] &= [0, 255(1-s)] \text{ for } s < 0.\end{aligned}$$

The quantization of $\hat{o}$ proceeds similar to the quantization of $\hat{s}$ with the only difference that the possible range of $\hat{o}$ is variable (it depends on $\hat{s}$). The offset value quantization costs another 8 flops leading to 13 flops for the whole quantization process.

In the final computation the collage error is evaluated. Since $E(D, R) = d_{D,R}(\hat{s}_q, \hat{o}_q)$, the collage error $E(D, R)$ can be computed via

$$\begin{aligned}d_{D,R}(\hat{s}_q, \hat{o}_q) &= \hat{s}_q(\hat{s}_q\langle D, D\rangle + 2(\hat{o}_q\langle D, \mathbf{1}\rangle - \langle R, D\rangle)) + \\ &\quad \hat{o}_q(\hat{o}_q n - 2\langle R, \mathbf{1}\rangle) + \langle R, R\rangle. \qquad (7)\end{aligned}$$

which requires another 12 flops. In summary, the computation of the quantized luminance parameters $(\hat{s}_q, \hat{o}_q)$ and the collage error $E(D, R)$ necessitates 35 flops, provided that the inner products are already available. [2]

We now look at how the calculation of the inner products $\langle D, R\rangle$, $\langle D, D\rangle$, $\langle D, \mathbf{1}\rangle$, $\langle R, R\rangle$, and $\langle R, \mathbf{1}\rangle$ is carried out in the encoder. When all range blocks are of the same shape and, thus, the canonical codebooks are identical for all ranges, the computations are organized in two nested loops:

Algorithm 1 (Standard computation)

```
• Global preprocessing: compute $\langle D, D\rangle, \langle D, \mathbf{1}\rangle$ for all codebook blocks $D$.
• For each range $R$ do:
    • Local preprocessing: compute $\langle R, R\rangle, \langle R, \mathbf{1}\rangle$.
    • For all codebook blocks $D$ do:
        • Compute $\langle D, R\rangle$ and $E(D, R)$.
```

When ranges have differing shapes, the computations of inner products of the types $\langle D, D\rangle$ and $\langle D, \mathbf{1}\rangle$ have to be moved to the local preprocessing since for each range the corresponding canonical codebook has to be employed.

[2] We note that a few of these flops can be saved, e.g., by assuming that $a = (\langle D, D\rangle - \langle D, \mathbf{1}\rangle^2/n)^{-1}$ is stored in place of $\langle D, D\rangle$, and $\langle D, \mathbf{1}\rangle$ is replaced by $b = \langle D, \mathbf{1}\rangle/n$. Then $\hat{s} = a \cdot (\langle D, R\rangle - b \cdot \langle R, \mathbf{1}\rangle)$, requiring only 3 flops instead of 7 as in (5). Also the last term $\langle R, R\rangle$ in (7) may safely be ignored since it is only an additive constant not depending on the codebook block D.

Since the calculation of $\langle D, R \rangle$ is in the innermost loop, it dominates the computational costs in the encoding. For large ranges this computation also contributes the major portion of the floating point operations required to calculate the collage error: the direct computation of $\langle D, R \rangle$ with R containing $16 \times 16 = 256$ pixels requires 511 floating point operations which is more than twelve times the cost of computing the error $E(D, R)$ when the inner products are known.

4.2 The proposed algorithm

Our lossless acceleration method takes advantage of the fact that the codebook blocks, taken from the downscaled image, are overlapping. The fast cross correlation —based on the cross correlation property of the Fourier Transform— is ideally suited to exploit this sort of codebook coherence.

The above analysis has indicated that the calculation of the inner products $\langle D, R \rangle$ between a codebook block D and a range R dominates the computational cost in the encoding. In order to determine for range R the optimal codebook block in the corresponding canonical codebook, one has to compute all the inner products $\langle D_{n_1,n_2}, R \rangle$, $0 \le n_1, n_2 < N/2$. These inner products $\langle D_{n_1,n_2}, R \rangle$ are nothing else but the cross correlation of the range R with the downscaled image. To make things explicit, we give formal definitions.

Definition 1 (Cross correlation) *The cross correlation $f_1 \circ f_2$ of images $f_1, f_2 \in \mathbb{R}^{K \times K}$, $K \in \mathbb{N}$, is defined by*

$$(f_1 \circ f_2)(n_1, n_2) = \sum_{k_1=0}^{K-1} \sum_{k_2=0}^{K-1} f_1(k_1, k_2) \cdot f_2((k_1 + n_1) \bmod K, (k_2 + n_2) \bmod K),$$

with $0 \le n_1, n_2 < K$.

Now let h denote the downscaled image of size $N/2 \times N/2$ pixels as defined in Section 2. Let the image $g \in \mathbb{R}^{N/2 \times N/2}$ be constructed by putting a range in the upper left corner and setting all other coefficients to zero; g is called a *zero-padded range*. For simplicity, we assume for the following formal definition that ranges do not wrap-around image borders; otherwise slight modifications in the definitions would be necessary. Assume that the pixels in the range have coordinates in the set $B \subset \{i_{top}, \dots, i_{bottom}\} \times \{j_{left}, \dots, j_{right}\}$, and the bounding box given by $i_{top}, i_{bottom}, j_{left}, j_{right}$ is of minimal size. When the size of the range is smaller or equal to $N/2$ in each dimension, i.e., when

$$i_{bottom} - i_{top} < N/2 \ \text{ and } \ j_{right} - j_{left} < N/2, \tag{8}$$

the zero-padded range $g \in \mathbb{R}^{N/2 \times N/2}$ is given by

$$g(i - i_{top}, j - j_{left}) = I(i, j), \text{ if } (i, j) \in B,$$

and $g(i, j) = 0$, otherwise. If equation (8) is not satisfied, one needs an appropriate folding of the range.

With the above definitions the cross correlation $(g \circ h)$ gives the inner products between a range and the codebook blocks of the corresponding canonical codebook via

$$(g \circ h)(n_1, n_2) = \langle D_{n_1,n_2}, R \rangle, \quad 0 \le n_1, n_2 < N/2.$$

The cross correlation can be carried out efficiently in the frequency domain.

Definition 2 (Discrete Fourier Transform, DFT) *Let $f \in \mathbb{R}^{K \times K}$ be an image. The Fourier Transform F of f is given by*

$$F(n_1, n_2) = \sum_{k_1=0}^{K-1} \sum_{k_2=0}^{K-1} f(k_1, k_2) W^{-n_1 k_1 - n_2 k_2},$$

where $W = \exp(2\pi \mathrm{i}/K)$, $0 \le n_1, n_2 < K$.

We build on the following well-known property of the Fourier Transform (cf. [15, p. 82]):

Theorem 3 *The Fourier Transform of the cross correlation $f_1 \circ f_2$ is given by $F_1^c * F_2$, where F_1^c denotes the complex conjugate of F_1 and $*$ stands for the Hadamard product.*

Since the Fourier Transform allows a very efficient implementation via the *Fast Fourier Transform* (FFT), the calculation of the cross calculation should be done in the frequency domain when the involved blocks are not too small.

The computation of the cross correlation between an image block R and the downscaled image of size $N/2 \times N/2$ is now organized as follows:

1. Compute the 2D-FFT of the downscaled image.

2. Enlarge image block R by zero padding to size $N/2 \times N/2$.

3. Compute the 2D-FFT of the enlarged image block.

4. Perform the complex conjugate of the previous transform.

5. Do the complex multiplication of both transforms.

6. Do the inverse FFT of the result.

Since one obtains the whole set $\langle D_{n_1,n_2}, R\rangle, 0 \leq n_1, n_2 < N/2$, at one batch, the outlined procedure takes the calculation of the inner products out of the inner loop in Algorithm 1 and places it into the local preprocessing. Moreover, the calculation of $\langle D, D\rangle, \langle D, \mathbf{1}\rangle$ for all codebook blocks of a canonical codebook requires a substantial amount of time but can be accelerated by the same cross correlation technique. The products $\langle D, \mathbf{1}\rangle$ are obtained by the cross correlation of the downscaled image with a 'range' where all intensities are set to unity (called the *range shape matrix*). The sum of the squares $\langle D, D\rangle$ is computed in the same way where all intensities in the downscaled image are squared before cross correlation. When all ranges are of the same shape, the algorithm for fractal encoding based on fast cross correlation can be summarized as follows:

Algorithm 2 (Fast cross correlation based fractal encoding)

A. Input: Image of size $N \times N$.

B. Global preprocessing.

1. *Compute the downscaled image by pixel averaging.*
2. *2D-FFT of the downscaled image.*
3. *Fast cross correlation with the range shape matrix yielding $\langle D, \mathbf{1}\rangle$ for all codebook blocks D.*
4. *Fast cross correlation of the squared downscaled image and the range shape matrix yielding the summed squares $\langle D, D\rangle$ for all codebook blocks D.*

C. For each range R do steps C1 to C4.

1. *Local preprocessing: Compute $\langle R, \mathbf{1}\rangle$ and $\langle R, R\rangle$.*
2. *Local preprocessing: Fast cross correlation of the range block using the result of step B2 obtaining all products $\langle D, R\rangle$.*
3. *For each codebook block D compute the (quantized) coefficients $\hat{s}_q, \hat{o}_q$, and the collage error $E(D, R)$ using the results of steps B3, B4, C1, and C2.*
4. *Extract the minimal collage error and output the fractal code for the range.*

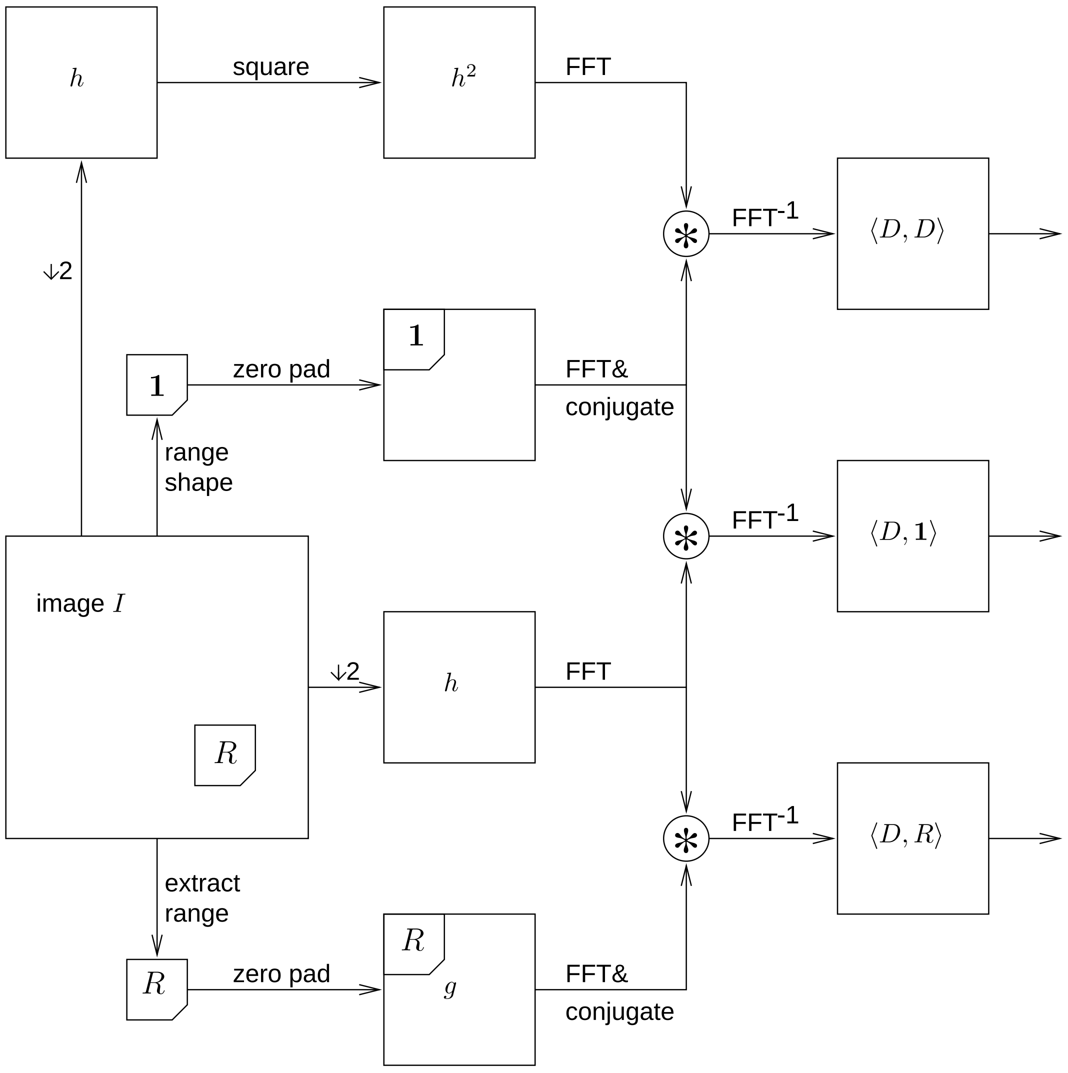


Fig. 2. Flow chart of the FFT-based technique for the computation of the arrays of the inner products $\langle D, D\rangle, \langle D, \mathbf{1}\rangle, \langle D, R\rangle$. The symbol $*$ denotes the Hadamard product of two complex Fourier coefficient matrices. Here the range block does not need to be of square shape.

In the case where many different range shapes are present in the image partition, the steps $B3, B4$ have to be moved to the local preprocessing. Figure 2 illustrates the part of the algorithm that computes the arrays of $\langle D, \mathbf{1}\rangle$, $\langle D, D\rangle$, and $\langle D, R\rangle$ with the FFT-based approach. Since we have only changed the way of *how* the inner products $\langle D, \mathbf{1}\rangle$, $\langle D, D\rangle$ and $\langle D, R\rangle$ are computed, we obtain the same results as for the case of direct computation, and, therefore, the acceleration method is lossless.

4.3 Implementation and results

To compare our proposed approach to the standard "brute force" method, we check the number of floating point operations that are required to compute the inner products $\langle D, R \rangle$ between a fixed range block R and all codebook blocks D. By doing this, one obtains speedup factors that are independent of a specific input. In the "brute force" case, an n-dimensional inner product takes $2n$ floating point operations (or $2n - 1$ when the summation does not start with zero). For an image of size $N \times N$, there are $N/2 \times N/2$ codebook blocks (we include blocks that wrap around image borders); thus, our total is

$$2n \cdot N/2 \cdot N/2 \text{ flops.} \tag{9}$$

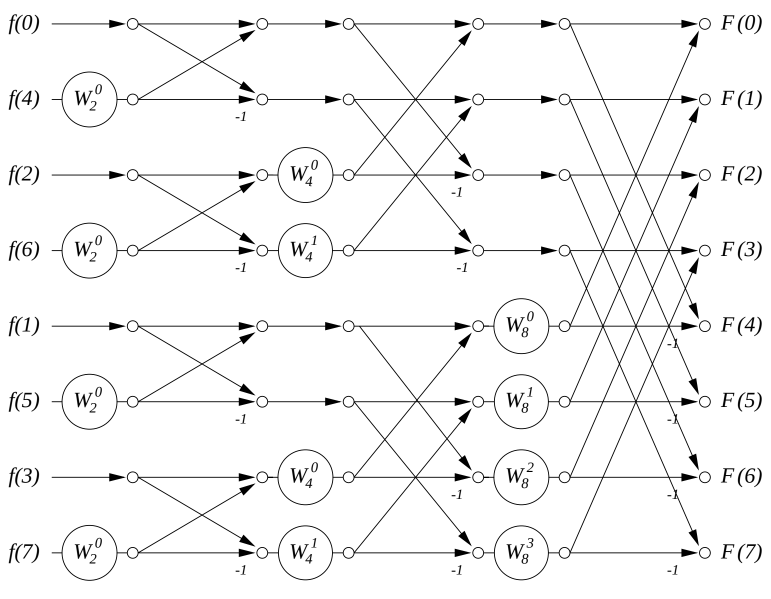
In the implementation of our new method we have built on the FFT code from [16]. It is a radix-2 decimation-in-time Fourier Transform. To handle the real input data, the data is assumed to be complex and after the forward complex transform the coefficients of the real transform are computed in the standard fashion (see [16]). With our FFT-based algorithm, we must take into account the costs for transforming the zero-padded range g, the multiplication between this transform and the Fourier Transform H of the downscaled image h, and the inverse transformation. Then the total number of flops to calculate the entire set of products $(\langle D_{n_1,n_2}, R \rangle)_{0 \le n_1, n_2 < N/2}$ for a given range R is given by

$$\frac{5N^2}{2} \log_2 N + 3N^2. \tag{10}$$

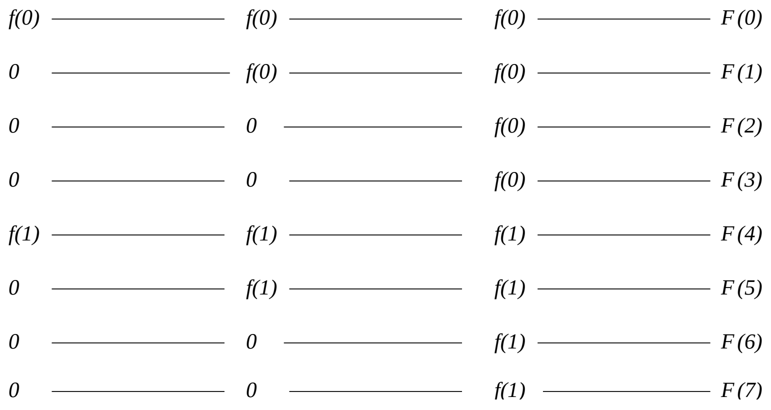
This number has to be compared with $n/2 \cdot N^2$ flops required by the standard computation for the same task, where n is the number of pixels in the range block R. This comparison indicates that the FFT-based technique is more efficient for range blocks with $n > 46$ pixels when the image size is $N = 256$, and for range blocks with $n > 51$ pixels when $N = 512$.

Furthermore, for the forward transform, performance can be enhanced even more because most entries in the zero padded range image are zero. As an example, the computation of an 8-point FFT on input data that has only two non-zero coefficients, namely $f(0)$ and $f(1)$, is shown in Figure 3. One can skip the computation of the first two stages and can directly compute the last stage. In general, when only the first k coefficients of a one-dimensional signal are non-zero, only $\lceil \log_2(k) \rceil$ stages have to be computed. Similar techniques have been used, e.g., to speed up the decoding procedure in DCT-based image coding schemes. Here, a large number of high frequency coefficients are zero, a fact that can be exploited by the decoder. For a discussion of this topic see, e.g., [13].

Table 1 presents the number of floating point operations required for computing the correlation between a single range block and all possible codebook blocks using *i)* the standard "brute force" method (cf. expression (9)), *ii)* the FFT-based approach (cf. expression (10)), and *iii)* the FFT-based approach



(a)



(b)

Fig. 3. (a) gives an 8-point radix-2 FFT, (b) shows the corresponding truncated FFT structure when most of the input values are 0 which is the case for our zero-padded range blocks.

with enhanced forward transform (floating point operations counted at run-time). The results show no speedup for small ranges of size 4×4. However, the speedup grows linearly with the number of pixels in the range. For the large 32×32 range size we get a speedup of about 30.

The actual timing results (CPU times) for the complete fractal coder show speedup factors that are only slightly smaller than the ones presented in Table 1 due to small overheads such as preprocessing. For example, in our implementation an acceleration factor of 7.67 was observed for total CPU time for

Table 1
Number of floating point operations (in thousands) to compute the correlations between a single range block and all possible codebook blocks using the standard "brute force" scheme, the fast FT, see expression (10), and the enhanced FFT exploiting zero padding (flops counted at run time); (a) for images of size 256×256, (b) for images of size 512×512.

	Floating Point Operations per Range Block (in 1000) Image Size 256×256			
Range Size	Standard Brute Force	Standard FFT	Enhanced FFT	Speedup
4×4	524	1507	929	0.56
8×8	2097	1507	976	2.15
16×16	8388	1507	1036	8.10
32×32	33554	1507	1139	29.46

(a)

	Floating Point Operations per Range Block (in 1000) Image Size 512×512			
Range Size	Standard Brute Force	Standard FFT	Enhanced FFT	Speedup
4×4	2097	6684	3837	0.55
8×8	8388	6684	4009	2.10
16×16	33554	6684	4193	8.00
32×32	134217	6684	4408	30.45

(b)

the case of encoding a 256×256 with a range partitioning of 16×16 blocks. The table shows a speedup of 8.10 for the dominating cross-correlation.

5 Features and extensions of FFT-based fractal image encoding

In this section we will consider further optimizations for our method and the use of isometries as well as the application of our method to highly irregular image partitions.

The radix-2 scheme we have employed for computing the FFT is suboptimal with respect to the number of arithmetic operations. The Fourier Transform for real-valued sequences with minimal number of arithmetic operations has been given in [24]. Since run-time optimization is highly machine-dependent we have instead chosen SGI's Complib library functions [5] that provide hardware-optimized FFT routines. Additional encoding time reductions of 30–50% have been observed. We have also considered *Number Theoretic Transforms* (NTT)[15]. The NTTs have a similar structure as a DFT but with the complex roots of unity replaced by integer roots of unity over a finite field. Those NTTs need only integer additions and shift operations, and, therefore, are less complex than a DFT. However, the restrictions imposed on image size and maximal intensity value appear to be too severe for our application.

It is easy to incorporate isometries of codebook blocks to enrich the codebook. In our approach we can take advantage of direct methods to obtain the Fourier Transforms for rotated and reflected images. Therefore, there is no need to compute 8 forward transforms for the 8 isometric versions of the downscaled image. Instead, one forward transform is computed and the Fourier Transform of the downscaled image is rotated and reflected directly in the Fourier domain. Experimental results show that the use of the eight isometric versions leads to an increase of encoding time by only a factor of five.

Our method has a strong potential in applications where an adaptive image partition provides for large irregularly shaped ranges and a fractal code is sought (see Figure 4 for an adaptive partition of the 512×512 image Lenna). Those partitions have been shown to lead to coding results that are among the best reported for fractal coders that are pure in the sense that they do not realize a hybrid approach by employing an image transformation before fractal coding [18]. Here, the direct computation of the inner products is considerably aggravated because of the irregular block shapes. Those irregular block shapes lead to additional costs since it is harder to decide whether or not a given pixel contributes to the inner product to be calculated. Also most other (lossy) acceleration techniques such as nearest-neighbor-search in a space of feature vectors [20], or classification [8] are not well suited because the heavy preprocessing costs must be paid for each range shape that occurs in the partition. A notable exception is the (lossy) multi-resolution technique in [14] which can be applied to irregular partitions without such a penalty.

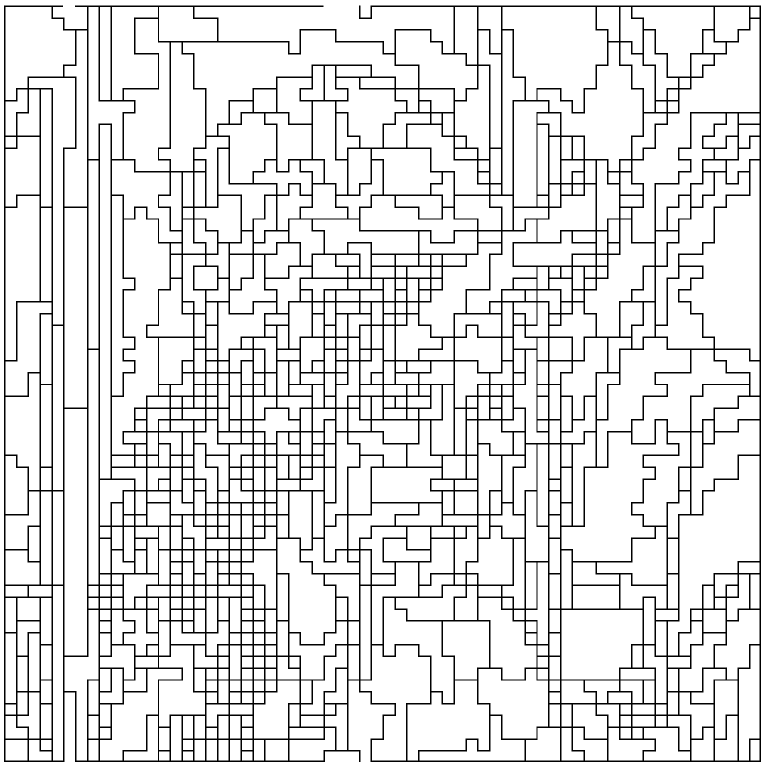


Fig. 4. Partition of the standard 512 × 512 Lenna image with 800 ranges.

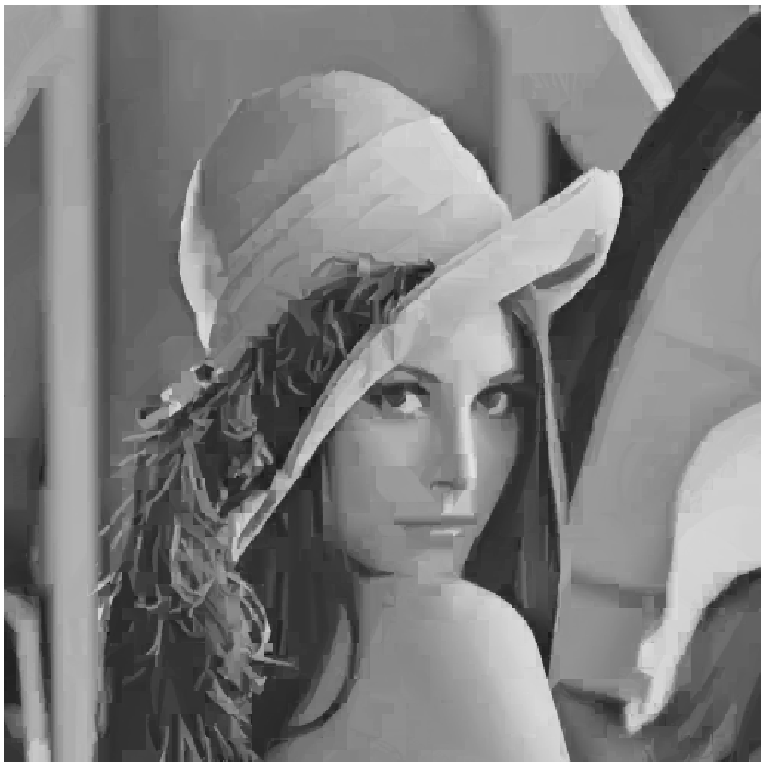


Fig. 5. Decoded image pertaining to the above partition, compression ratio 68:1, 29.73 dB PSNR.

The FFT-based approach handles the case of an irregular range shape with ease by zero-padding the pixels that are not in the range. When ranges have different shapes, the computation of $\langle D, \mathbf{1}\rangle, \langle D, D\rangle$ cannot be done as a preprocessing step anymore as it is the case for uniform partitions. Therefore, we obtain an additional gain with our method. Figure 4 shows an irregular adaptive partition which we obtained using the methods described in [18]. For this partition the speedup obtained with the FFT-based technique as compared to a fully optimized direct approach is about 18 for the radix-2-based version and 35 for the Complib version. Figure 5 shows the decoded image pertaining to the partition of Figure 4 and the fractal code determined using our approach.

6 Conclusions

In this paper we have shown how the FFT-based cross correlation provides a new approach to reducing the time complexity of fractal image encoding. The time reduction is achieved in a lossless fashion, i.e., one does not sacrifice coding quality for the sake of speedup. We have presented experimental results for various image sizes, range sizes, and optimization levels showing the benefits of the new approach as compared to the direct method. The computing time per range depends linearly on the number of pixels in the range. Speedup factors of about 8 for ranges with 256 pixels and factors of 30 for ranges with 1024 pixels were observed. The proposed approach is particularly well suited for applications where an adaptive image partition provides for large irregularly shaped ranges and where a fractal code is sought.

References

[1] M. Barnsley, L. Hurd, Fractal Image Compression, Academic Press, San Diego, 1988

[2] K.-U. Barthel, J. Schüttemeyer, T. Voyé, P. Noll, "A new image coding technique unifying fractal and transform coding," Proc. IEEE Int. Conf. on Image Processing, Austin, Texas, 1994, pp.III:112–116

[3] T. Bedford, F.M. Dekking, M. Breewer, M.S. Keane, "Fractal coding of monochrome images," Signal Processing, vol. 6, 1994, pp.405–419

[4] G. Caso, P. Obrador, C.-C.J. Kuo, "Fast methods for fractal image coding," Proc. SPIE Visual Communication and Image Processing, vol. 2501, 1995, pp.583–594

[5] CHALLENGE-complib 2.2 for IRIX 6.2, Silicon Graphics Scientific Mathematical Library, Silicon Graphics, Inc.

[6] G. M. Davis, “A wavelet-based analysis of fractal image compression,” Trans. on Image Processing, vol. 7, no. 2, Feb. 1998

[7] M. Dekking, “Fractal image coding: some mathematical remarks on its limits and its prospects,” in: Y. Fisher, ed., Conf. Proc. NATO ASI Fractal Image Encoding and Analysis, Trondheim, 1995, Springer-Verlag, New York, 1998

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

[9] Y. Fisher, “Fractal image compression with quads,” in [8]

[10] R. Hamzaoui, “Codebook clustering by self-organizing maps for fractal image compression,” in: Fractal Image Encoding and Analysis, Nato ASI, Trondheim, July 1995, Fractals, Supplementary Issue, vol. 5, April 1997

[11] A.E. Jacquin, “Image coding based on a fractal theory of iterated contractive image transformations,” IEEE Trans. Image Processing, vol. 1, 1992, pp.18–30

[12] H. Krupnik, D. Malah, E. Karnin, “Fractal representation of images via the discrete wavelet transform,” Proc. IEEE Conf. of Electrical Engineering, Tel Aviv, March 1995

[13] K. Lengwehasatit, A. Ortega, “Distortion/decoding time tradeoffs in software DCT-based image coding,” Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Munich, 1997, pp.IV:2725–2729

[14] H. Lin, A.N. Venetsanopoulos, “A pyramid algorithm for fast fractal image compression,” Proc. IEEE Int. Conf. on Image Processing, Washington D.C., 1995, pp.III:596–599

[15] H.J. Nussbaumer, Fast Fourier Transform and Convolution Algorithms, Springer-Verlag, Berlin, 1990

[16] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C, 2nd edition, Cambridge University Press, 1992

[17] M. Ramkumar, G.V. Anand, “An FFT-based technique for fast fractal image compression,” Signal Processing, vol. 63, 1997, pp.263–268

[18] M. Ruhl, H. Hartenstein, D. Saupe, “Adaptive partitions for fractal image compression,” Proc. IEEE Int. Conf. on Image Processing, Santa Barbara, 1997, pp.II:310–313

[19] D. Saupe, “A new view of fractal image compression as convolution transform coding,” IEEE Signal Processing Letters, vol. 3, no. 7, July 1996, pp.193–195

[20] D. Saupe, “Fractal image compression via nearest neighbor search,” in: Y. Fisher, ed., Conf. Proc. NATO ASI Fractal Image Encoding and Analysis, Trondheim, 1995, Springer-Verlag, New York, 1998

[21] D. Saupe, H. Hartenstein, “Lossless acceleration of fractal image compression by fast convolution,” Proc. IEEE Int. Conf. on Image Processing, Lausanne, 1996, pp.I:185–188

[22] D. Saupe, R. Hamzaoui, "Complexity reduction methods for fractal image compression," in: J.M. Blackledge, ed., I.M.A. Conf. Proc. on Image Processing; Mathematical Methods and Applications 1994, Oxford University Press, 1997, pp.211-221

[23] D. Saupe, S. Jacob, "Variance-based quadtrees in fractal image compression," Electronics Letters, vol. 33, no. 1, 1997, pp.46–48

[24] H. Sorensen, D. Jones, M. Heidman, S. Burrus, "Real-valued Fast Fourier Transform algorithm," IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-35, 1987, pp.849–863