Image Compression
Based on a Fractal Theory

S701, 18 January 1994
Carsten Frigaard, Jess Gade, Thomas T. Hemmingsen and Torben Sand

Institute for Electronic Systems
Aalborg University, Denmark

Email: cfri91@mcenroe.control.auc.dk

Abstract *Data compression has become an important issue in relation to storage and transmission of information. Specifically digital image compression is important due to the high storage and transmission requirements. Various compression methods have been proposed in recent years using different techniques to achieve high compression ratios. All these methods share the same characteristic of being approximate, i.e. the compressed images will be approximations of the originals.*

A relatively new approach is to construct compression algorithms by exploiting the theoretical foundation given by fractal theory. The basic principle is that an image can be reconstructed by using the self similarities in the image itself. When encoding an image, the algorithm partitions the image into a number of square blocks (domain blocks). After this a new partition into smaller blocks (range blocks) takes place. For every range block the best matching domain block is searched among all domain blocks by performing a set of transformations on the blocks. The compression is obtained by storing only the descriptions of these transformations.

A crucial point in the encoding procedure is to be able to select only the "best matching" domain blocks in advance, by an efficient classification and only compute the transformations on these blocks. This is done exclusively to reduce the encoding time which would be unacceptable high otherwise. When decoding the image by continual iteration, the original image will be reconstructed with some approximation.

In this paper we examine and describe an implementation of a fractal compression method proposed by [1], and state our improvements of the compression scheme. The results obtained will be compared with the original proposal and standard compression techniques. Finally, suggestions of further improvements and use of the method in other areas will be presented.

1 Introduction

Digital images posses the characteristics of being both data extensive and relatively redundant objects which makes them an obvious target for compression.

The acquisition of the code can be regarded as a direct reduction in costs, due to the requirements of often expensive storage space of uncompressed images.

The focus in this paper is upon examination and improvement of a digital image compression method, based on a fractal theory of iterated contractive transformations as proposed by [1]. In this method, the redundancy of the digital image is assumed to be exploitable through self-transformability on a block-wise basis, i.e. the image can be built of transformed copies of parts of itself.

A digital image will as a result of the encoding phase be represented by a set of transformations, the *fractal code*, which contains the complete information needed to reconstruct the image. In the decoding process, an approximation of the original image will emerge, when the set of transformations are iterated on any initial image.

The compression scheme proposed by [1] will be improved by an enhanced block classification routine. In addition, we suggest further applications of the compression scheme.

In the method section, we will present a revised deduction of both the encoding and decoding procedures, as well as a suggestion of a novel block classification routine.

In the results section, we state the results obtained by the improved compression scheme on some typical images. In addition, we will compare the aspects of the compression scheme to a standard image compression method.

In the discussion section, we will draw conclusions on the overall performance of this compression scheme as well as its advantages and disadvantages.

2 Method

In this section we will present a review of the fundamentals of the compression scheme, i.e. the theoretical foundations of the method, the general design of a coding system and the principle of the computation of the fractal code, all proposed by [1]. In addition, we will present our suggestions of improvements of the overall compression scheme.

2.1 Theoretical foundations

Let (M, δ) denote a metric space of digital images, where δ is a given metric – distortion measure, and let μ_{orig} be an original image we want to encode. The *inverse problem* of iterated transformation theory is the construction of a contractive image transformation τ, defined from the space (M, δ) to itself.

The requirements on the transformation τ are defined as follows:

$$\exists s < 1 \text{ such that } \forall \mu, \nu \in M$$
$$\delta(\tau(\mu), \tau(\nu)) \leq s\delta(\mu, \nu) \quad (1)$$

and

$$\delta(\mu_{\text{orig}}, \tau(\mu_{\text{orig}})) \text{ is as "small" as possible.} \quad (2)$$

The scalar s is called the *contractivity* of τ. Under these conditions, and provided that τ requires less storage space than the original image, τ is a *lossy image code* for μ_{orig} [1].

2.2 Distortion measure

In the following μ is an image or a block and $\tilde{\mu}$ is the approximation of μ. A square block of size $B \times B$ pixels located at position (i_0, j_0) in μ is denoted by $S(i_0, j_0, B)$ or $\mu_{\restriction S}$.

To measure the distortion between the two blocks μ and $\tilde{\mu}$ of sizes $B \times B$ pixels, the following *signal-to-noise ratio* is used:

$$\text{SNR} = 10 \log_{10} \left(\frac{dr(\mu)^2}{\delta_{L_2}(\mu, \tilde{\mu})/B^2} \right) \quad (3)$$

where $dr(\mu)$ is the dynamic range of μ, that is the maximum grey level difference between two pixels in the block. $\delta_{L_2}(\mu, \tilde{\mu})$ is the *squared* distortion between the two blocks:

$$\delta_{L_2}(\mu, \tilde{\mu}) = \sum_{0 \leq i,j < B} (\mu_{i,j} - \tilde{\mu}_{i,j})^2 \quad (4)$$

The *mean squared* distortion is also used in the quadtree partitioning scheme (see section 2.6). Other metrics could be used [2]

$$\delta_{sup}(\mu, \tilde{\mu}) = \sum_{0 \leq i,j < B} |\mu_{i,j} - \tilde{\mu}_{i,j}| \quad (5)$$

$$\delta_{rms}(\mu, \tilde{\mu}) = \sqrt{\int_{B^2} (\mu_{i,j} - \tilde{\mu}_{i,j})^2} \quad (6)$$

but the latter has more complicated contractivity requirements.

2.3 Encoding

First the image is partitioned into a number of blocks of size $2B \times 2B$ pixels called domain blocks, which are used as building blocks, and a number of blocks of size $B \times B$ pixels referred to as range blocks. For each range block the aim is to find the domain block and transformation which is the best match of the original range block (see figure 2.2).

The transformation τ_i is defined as the transformation from domain block D_i to range block R_i and is written as the composition of two transformations:

$$\tau_i = T_i \circ S_i$$

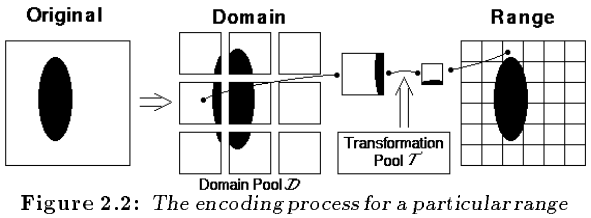


Figure 2.2: *The encoding process for a particular range block.*

where S_i and T_i are the so-called *geometric* and *massic* parts of τ_i, respectively.

A pool of domain blocks $\mathcal{D}$ is defined and consists of all image blocks of size $2B \times 2B$ which can be extracted from the original image, and a pool of massic transformations $\mathcal{T}$, made of all block transformations T_i. The problem of finding the best match is then equal to the problem of finding the "best pair" $(D_i, T_i) \in \mathcal{D} \times \mathcal{T}$, which complies with the distortion

$$\delta_{L_2}(\mu_{\restriction R_i}, T_i \circ S_i(\mu_{\restriction D_i})) \text{ is minimum.}$$

This is done in the following way:

If the range block is classified as a shade block (see section 2.5 for details of the classification) the transformation only has a massic part: absorption at grey level g_0:

$$(\theta\mu)_{i,j} = g_0$$

As this transformation always generates a uniform shade block, there is no need to search for the best matching domain block and the transformation therefore contains no geometric part.

If the range block is not a shade block the transformation consists of both a geometric and a massic part:

1. *Geometric part:* A down scaling of the domain block in size from $2B \times 2B$ pixels to $B \times B$ pixels and a geometric displacement from the location of the domain block to the location of the range block.

2. *Massic part:* Two transformations which manipulates the grey levels of the block and an isometric transformation which simply shuffles pixels within the block:

   (a) Contrast scaling by α:

   $$(\sigma\mu)_{i,j} = \alpha\mu_{i,j}$$

   (b) Luminance shift by Δg:

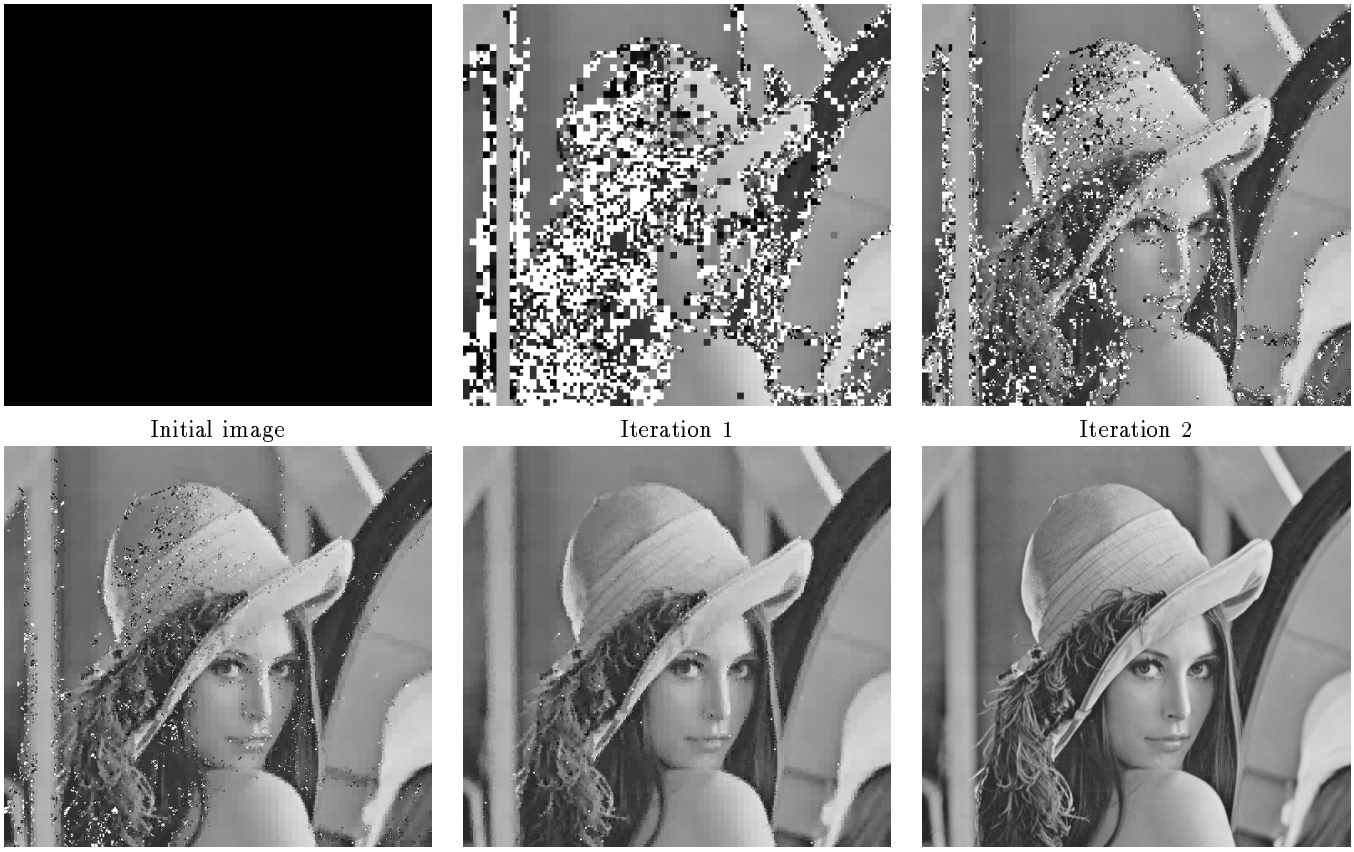
   $$(\tau\mu)_{i,j} = \mu_{i,j} + \Delta g$$

   (c) Isometrie by ι_i:

   The isometries consists of one of eight different transformations shown in table 2.1.

The total transformation from a domain block $\mu_{\restriction D_j}$ to a range block $\mu_{\restriction R_i}$ is then

$$T_i(S_j(\mu_{\restriction D_j})) = \iota_i(\alpha_i(S_j(\mu_{\restriction D_j})) + \Delta g_i) \quad (7)$$



Initial image | Iteration 1 | Iteration 2

Iteration 3 | Iteration 15 | Original 'Lenna'

Ratio=11.31, SNR=25.95dB

Figure 2.1: *Iterations of fractal code for 'Lenna' on initial black image.*

With the δ_{L_2} metric we are able to calculate the contractivity of a transformation - for example the contrast scaling on the block $S(i_0, j_0, B)$

$$\begin{aligned} &\delta_{L_2}\left(\sigma\left(\mu_{\restriction S}\right), \sigma\left(\nu_{\restriction S}\right)\right) \\ &\quad = \sum_{0\le i,j<B}\left(\alpha\mu_{i_0+i,j_0+j} - \alpha\nu_{i_0+i,j_0+j}\right)^2 \\ &\quad = \alpha^2\delta_{L_2}\left(\mu_{\restriction S}, \nu_{\restriction S}\right) \end{aligned} \tag{8}$$

so the contractivity of this transformation is α^2.

2.4 Decoding

Given the fractal code of an image. The reconstruction of this image is done by iterating the transformations on an arbitrary initial image. As the decoding proceeds the image converges to a stabile image, as seen on figure 2.1.

The convergence to a stable image, disregarding the initial image μ_0, can be stated by

$$\begin{aligned} \delta\left(\mu_{\text{orig}}, \tau^n\left(\mu_0\right)\right) \quad \le \quad & \frac{1}{1-s}\delta\left(\mu_{\text{orig}}, \tau\left(\mu_{\text{orig}}\right)\right) \\ & + s^n\delta\left(\mu_{\text{orig}}, \mu_0\right) \end{aligned} \tag{9}$$

where the metric δ in our case is δ_{L_2}. This result can be proven by use of the triangular inequality [3]. The convergence of course requires that the contractivity s is less than

Table 2.1: *The eight isometric transformations.*

No.	Function	Formula
1	Identity	$(\iota_1)_{i,j} = \mu_{i,j}$
2	Reflection about mid-vertical axis	$(\iota_2)_{i,j} = \mu_{i,B-1-j}$
3	Reflection about mid-horizontal axis	$(\iota_3)_{i,j} = \mu_{B-1-i,j}$
4	Reflection about first diagonal	$(\iota_4)_{i,j} = \mu_{j,i}$
5	Reflection about second diagonal	$(\iota_5)_{i,j} = \mu_{B-1-j,B-1-i}$
6	Rotation, through $+90°$	$(\iota_6)_{i,j} = \mu_{j,B-1-i}$
7	Rotation, through $+180°$	$(\iota_7)_{i,j} = \mu_{B-1-i,B-1-j}$
8	Rotation, through $-90°$	$(\iota_8)_{i,j} = \mu_{B-1-j,i}$

one. However It is not required that every block transformation is contractive but the set of transformations must be eventually contractive [2]to ensure the convergence of the image.

2.5 Classification

The optimal encoding of an image, and hence the optimal quality of the decoded image, requires the computation of the cartesian product of transformations and available domain blocks, i.e. $\mathcal{T} \times \mathcal{D}$. This, however, arises an unacceptable high encoding time, due to the extensive amount of computations the encoder must perform.

In order to reduce the encoding time, it is important to be selective regarding the set of domain blocks in which the best match to a specific range block is to be found. The idea is hence that for a particular range block, we only want to search for the best matching domain block among those domain blocks which posses the same characteristics as the range block. This will be called a classification of the image blocks. Both range and domain blocks must be submitted to the same classification algorithm in order to be able to compare blocks of different sizes.

Numerous classification schemes exist. In the VQ technique [4], edge blocks are examined, i.e. blocks in which a distinct edge appears. The argument for this approach is that the edges are the more determining elements of the quality of the image.

The classification in [1] is executed by categorizing the individual blocks into one of three sets: edges, shades and midranges. Edge blocks are blocks which contain a "distinct" edge, although this is determined by a subjective threshold. Shade blocks contain no distinctive features. Midrange blocks posses some structure, although no specific edge is present. The partitioning of the image blocks into three categories thereby reduces the encoding time (although this is highly determined by the individual image).

Feature Space

A new approach to the classification problem is now presented. The concept of a *feature space* is that the type of a block will be measured in a continuous scale, in contrast to the traditional way of classifying a block into one of a certain number of discrete categories, e.g. "edges, shades and midranges".

A block will be submitted to one or several algorithms from which the outputs, respectively, will be a continuous measurement of some feature/parameter in the block. The block will then be represented as a coordinate/vector in an n-dimensional space, where n is the number of parameters/features.

The search for the best matching domain block to a given range block can then be restricted to an examination of the domain blocks with coordinates in a narrow area around the coordinate of the range block. "Narrow" can be subjectively determined. If the search area is extended it implies a greater probability of finding a better matching block, giving a higher SNR, but at a cost of increased encoding time. However, the quality of the classification, i.e. the choice of parameters to be computed, would, in theory, assure that only a relatively small number of domain blocks around the coordinate of the range block would have to be examined.

The right choice of parameters is, however, seemingly difficult to determine.

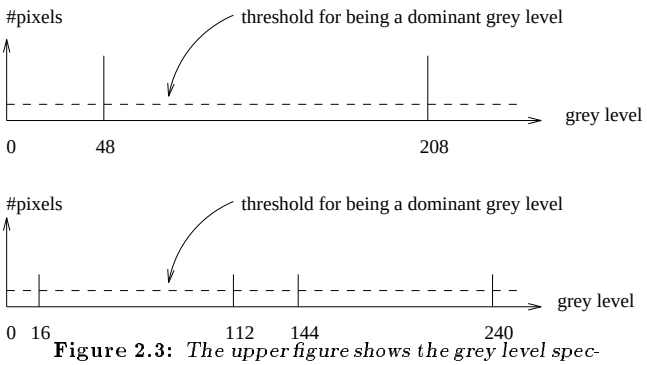


Figure 2.3: *The upper figure shows the grey level spectrum of a block which could contain an edge. The number of dominant grey levels is two. In the lower figure, the grey level spectrum of a block with same grey level standard deviation is shown. However, the represented lower block has, in contrary to the upper block, four dominant colors*

Parameters used in the Feature Space

The first parameter is the *grey level standard deviation* of pixels within a block. This gives a fairly precise classification of the different types of image blocks, although this measurement does not consider the fact that the grey level std. deviation of a block can be altered due to the contrast scaling and luminance shift transformations. It is theoretically possible to find a better matching block at an entirely different location on the std. deviation scale.

The grey level standard deviation of a block is computed in the following way:

$$\sigma_c = \sqrt{\frac{1}{P}\sum_{i=0}^{P} p_i^2 - \left(\frac{1}{P}\sum_{i=0}^{P} p_i\right)^2} \qquad (10)$$

where p_i is the grey level of pixel number i and P is the number of pixels in a block

The second choice of parameter is the *number of dominant grey levels* within a block. This measurement ensures a further partitioning of different blocks with the same grey level std. deviation. Consider figure 2.3, where the grey level spectrum of two ad hoc blocks are presented. The two blocks posses the same grey level std. deviation although being entirely different. The first could be an edge block with two dominant grey levels. The second could be more varying with four dominant grey levels.

The choice of threshold for being a dominant grey level is subjectively determined.

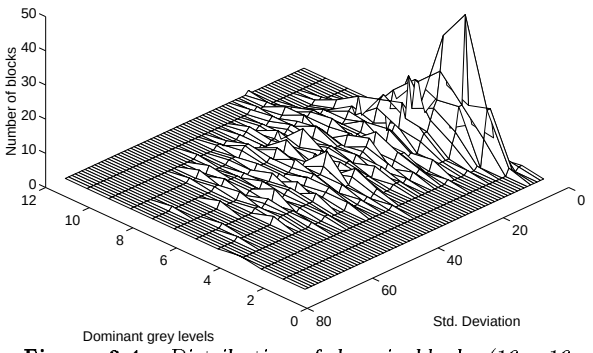


Figure 2.4: *Distribution of domain blocks (16 · 16 pixels) in feature space of "Lenna"*

The number of dominant grey levels of a block is computed in the following way:

- If the number of pixels of a certain grey level exceeds a threshold then increase the number of dominant grey levels for the block.

It is clear that this algorithm do posses some weakness. Imagine the distribution of grey levels for a block, which consists of many small and closely tied grey level contributions, none of which exceeds the threshold value. The joint effect of these independent grey level contributions appears to be equal to a block, where only one grey level value is present (in the local area we are regarding). The number of dominant grey levels is one less for the first block than for the second.

As an example, the distribution of the coordinates in the feature space for Lenna is shown in figure 2.4. A relatively large portion of the 16×16 pixel domain blocks manifests itself in the low end of the grey level std. deviation axis, with a number of dominant grey levels in the range from two to eight. This relates to the set of shade blocks.

Even with the new classification terminology, we still refer to the nature of a block as being either a shade block or a non-shade block, where the threshold is subjectively determined. In figure 2.4, the large "hump" consists of shade blocks.

2.6 Quad-tree partitioning

The size of the range blocks is an important parameter in the encoding scheme. An encoding with small range blocks ($B < 8$) results in a high SNR but a low compression ratio. With a higher B, the compression is increased but the details in the image are lost. To capture fine details while preserving a high compression ratio, a *quad-tree* partitioning scheme is introduced under the encoding phase.

The encoding is done in the following way: The encoding of the large range blocks is done as described earlier. The encoded range block is then divided into four sub blocks of size $B/2 \times B/2$, and the distortion measures δ_{L_2} are computed between these and the corresponding sub blocks in the original image. For each sub block, where the distortion exceeds a threshold value, a transformation for the sub block is found and appended to the main transformation. If either three or four sub transformations are needed the main transformation is discarded and the block is encoded as four sub transformations. The partitioning scheme can then be used again on each of the encoded sub blocks, allowing the encoding to take place at n levels.

The quad-tree partitioning allows the encoding scheme to take advantage of using large range blocks in smooth regions of the image, and smaller range blocks in more detailed parts of the image. This procedure also has the positive property of being easy to implement with a fully recursive approach.

2.7 Entropy coding

Since the fractal code for the encoding of a given image tends to contain a great amount of redundancy, an entropy coding of this code will be profitable. An entropy coding will also help to eliminate any inefficiency in the storage of the fractal code due to either an insufficient fractal code format or to special image properties, like a completely uniform image.

A simple entropy coding is the Shannon-Fano encoding algorithm [5], which rely on sorting q_i different messages with respect to their probabilities $p_{q_1}, p_{q_2}, ...p_{q_i}$ in descending order. We call F_i the cumulated probability and it is defined such that:

$$\begin{array}{rcll} F_i & \stackrel{def}{=} & 0 & \text{for } i = 1 \\ F_i & = & \sum_{k=1}^{i-1} p_{q_k} & \text{for } i \neq 1 \end{array} \tag{11}$$

To represent a given message q_i, we can compute the necessary number of bit, n_i, using:

$$\log_2 \left(\frac{1}{p_{q_i}} \right) \leq n_i < 1 + \log_2 \left(\frac{1}{p_{q_i}} \right) \tag{12}$$

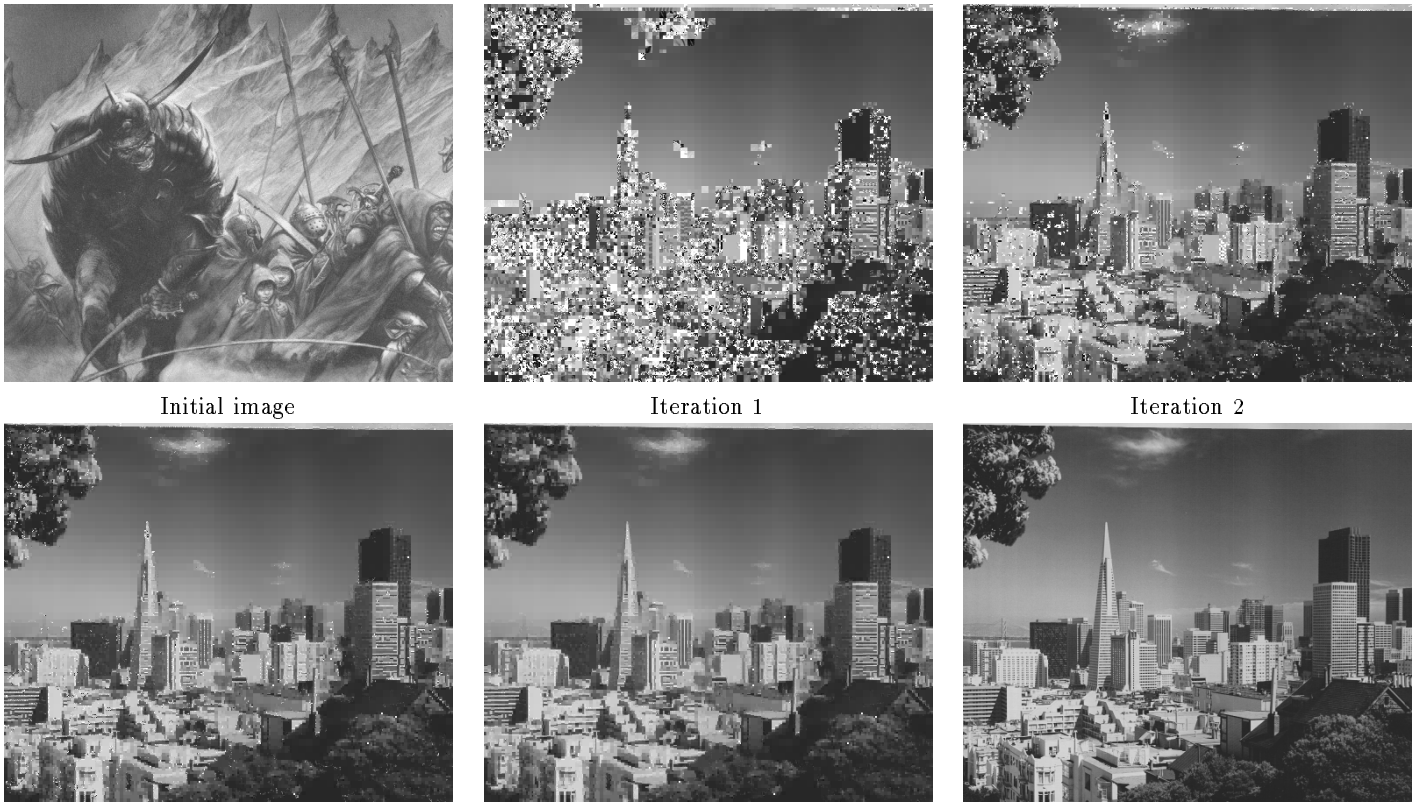
and the encoded output c_i of a given message q_i is then given by the binary fraction of F_i, truncated to the first n_i bit:

$$c_i = (F_i)_{\text{binary } n_i \text{bit}} \tag{13}$$

where the binary fraction is given by:

$$.b_1 b_2 b_3 ... b_k = \frac{b_1}{2} + \frac{b_2}{2^2} + \frac{b_3}{2^3} + \cdots + \frac{b_k}{2^k} \tag{14}$$

To be able to reconstruct the decoded messages, it is necessary to store the encoded message c_i along with a header containing information about the original message's probability p_{q_i}.



Initial image | Iteration 1 | Iteration 2

Iteration 3 | Iteration 15 | Original 'San Francisco'

Ratio=7.34, SNR=21.18dB

Figure 3.1: *Iterations of fractal code for 'San Francisco' on initial 'Orc' image.*

3 Results

Throughout these tests the original 512×480, 8-bpp 'Lenna' and 432×512, 8-bpp 'San Francisco' images were used. If nothing else is noted, a two-level quad-tree partitioning was used with domain blocks of size 16×16 and 8×8 pixels and corresponding range blocks at size 8×8 and 4×4 pixels, respectively. The domain pool was found by sliding a window over the original image using a step of four pixels in both x- and y-direction. A maximum of ten domain blocks was used to find the best match in the feature space for any range block. The fractal code is decoded using 15 iterations on any initial image.

A figure showing the decoding of 'Lenna' on an initial black image is presented in figure 2.1. Figure 3.1 shows the decoding of 'San Francisco' based on an initial 'Orc' image. The sequences demonstrates that the iterations converges to a stable image, disregarding the initial image. Figure 3.2 illustrates the convergence of the iterations in terms of SNR for 'Lenna' and 'San Francisco' based on the two different initial images. It is required to iterate approximately six times in order to reach a stable image. This number remains virtually unchanged disregarding which fractal code is iterated upon and which initial image is used.

3.1 System performance

A direct measurement of the performance of the fractal compression method compared to a standard compression technique (JPEG) was conducted by plotting the SNR corresponding to different compression ratios for both compressors. The plot is presented in figure 3.3. The result obtained by [1] is also presented for reference.

We observe that for compression ratios below 50:1 JPEG is better than fractal compression, but for compression ratios above the 50:1 breakpoint fractal compression performs better with respect to SNR.

For comparison figure 3.6 shows the output of both compressors when the compression ratio is 50:1. Here we notice how JPEG is able to obtain its large compression ratio by reducing the number of grey levels used and by merging uniform areas into large blocks. The fractal compression preserves the range of 256 grey levels but tends to miss the overall structure of 'Lenna'. This difference clearly illustrates the diversity between JPEG and fractal compression.

To investigate the quality of the classification, figure 3.4 illustrates the relation between the number of domain blocks examined for each range block compared with SNR. The result shows that it is sufficient to compute the transformations on approximately ten of the closest domain

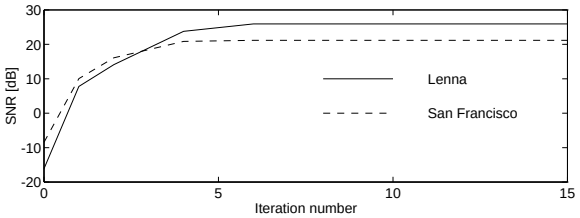


Figure 3.2: *Convergence to a stable image in terms of SNR when decoding 'Lenna' on an initial black image and 'San Francisco' on an initial 'Orc' image.*

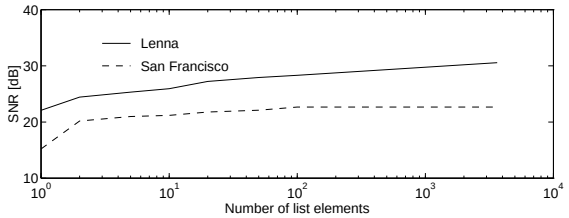


Figure 3.4: *The effect of varying the number of domain blocks to be compared with a given range block.*

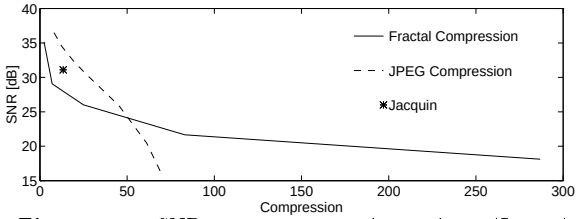


Figure 3.3: *SNR versus compression-ratio on 'Lenna' for both the fractal- and JPEG-compressor.*

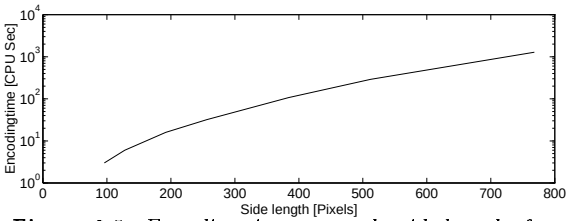


Figure 3.5: *Encoding time versus the side length of an image. The image was encoded on a Sun Sparc station.*

blocks in the feature space, since no significant improvement of SNR is observed for values above this number. This indicates that the feature space concept is a major improvement of the traditional classification schemes. However it is not certain that the features chosen are optimal.

Complexity

The original encoding scheme has a worst case computational complexity of $\approx n^4$ in the encoding phase (n is the side length of the image). This complexity remains unchanged despite the new classification scheme described in this paper. However the sufficient number of domain blocks to be examined for every range block has been greatly reduced, resulting in a significantly lower encoding time.

The complexity can be described by the following equation;

$$c(n) \approx k_1 n^2 + k_2 n^4 \qquad k_1 \gg k_2 \tag{15}$$

The factor $k_1 n^2$ origins from constant time calculations that are performed when the best matching domain blocks are found for each range block in the image. In order to find the domain blocks to be compared with each range block, the algorithm has to search through a list of the domain blocks that increases in size proportional to n^2. This results in the factor $k_2 n^4$. Because the list search is far less time consuming compared to the constant time calculations when encoding 'normal sized' (i.e. smaller than 1024×1024 pixels) images, the practical complexity is smaller than the theoretical.

Measurements of encoding time versus the side length of the encoded image are listed in figure 3.5. The complexity is from the data in the figure calculated to be $\approx n^3$ for images smaller than 1024×1024 pixels.

3.2 Fractal code analysis

Partitioning and error of 'Lenna'

A two-level partitioning of 'Lenna' in shades and non-shades is shown in figure 3.7. Here we notice that the algorithm captures uniform areas with few details using larger blocks whereas the image is divided into smaller blocks when the image is more detailed. This image adaptive feature increases both the theoretical compression rate and the image quality but is mostly useful for partitioning in three levels or less.

An enhanced error image for the decoded 'Lenna' is shown in figure 3.7. The error ϵ is calculated as:

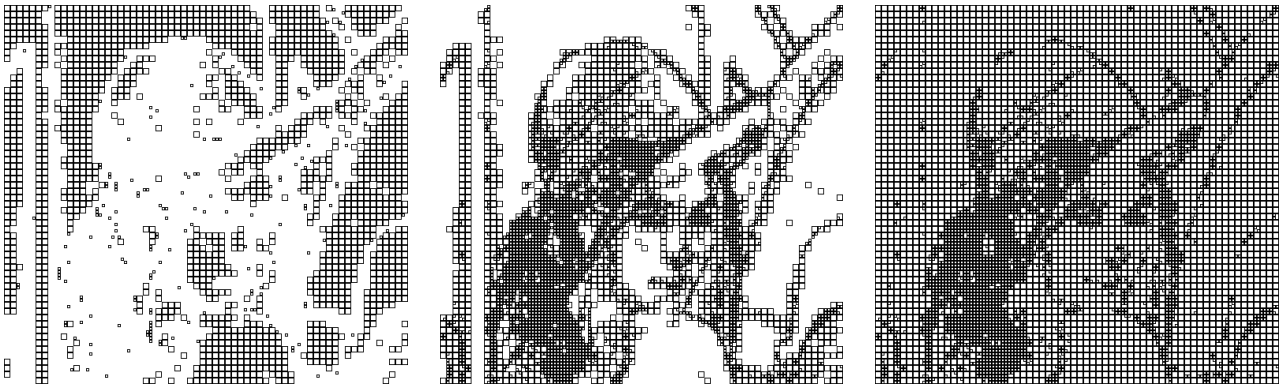
$$\epsilon = 5 \cdot |\mu_{i,j} - \tilde{\mu}_{i,j}| \tag{16}$$

The image reveals that errors occur mostly at sharp edges or in areas with detailed texture.

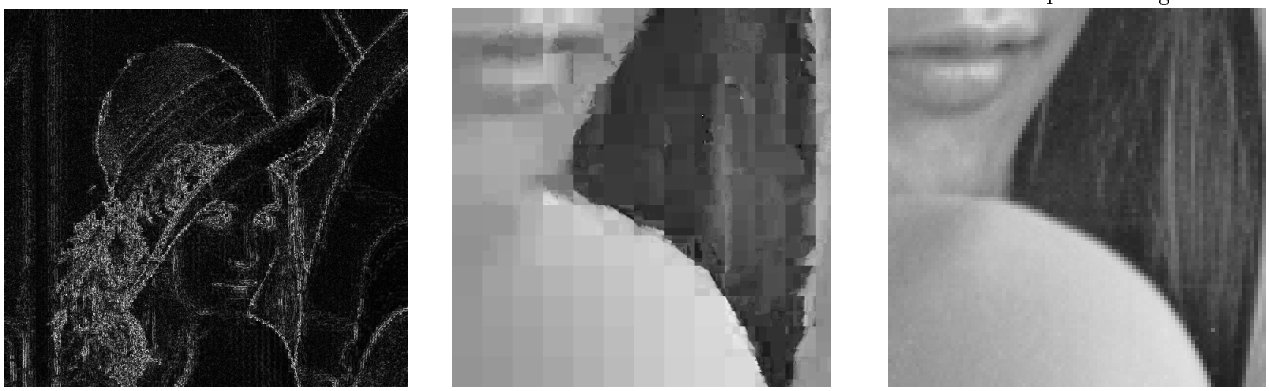
The zoom in figure 3.7, which shows the magnified shoulder of 'Lenna', illustrates the fractal nature of the scheme and although errors occur, the reconstruction is free of edge degradation in the form of a 'staircase effect'. Other fine details like the feathers in the hat tends to be smoothed out. The block structure is clearly visible which is a direct result of the block copying nature of the scheme.

The distribution of parameters in the fractal code

To make an analysis of the distribution of different parameters in the fractal code for 'Lenna' these parameters are plotted with respect to their relative probability in

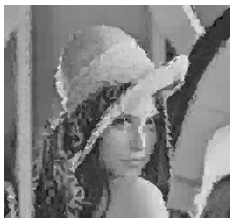


Shade blocks | Non shade blocks | Total partitioning



Enhanced Error image | Zoom on shoulder of encoded 'Lenna' | Zoom on original 'Lenna'

Figure 3.7: *The quad-tree partitioning of 'Lenna' with use of two levels plus an enhanced error image. The zoomed images illustrates the fractal nature of the scheme. The zooms used 16 · 16 domain blocks and 8 · 8 pixels range blocks.*



Fractal compression
Ratio=67.92:1 SNR=21.76dB

JPEG compression
Ratio=61.26:1 SNR=20.45dB

Figure 3.6: *'Lenna' compressed with fractal compression and JPEG. The fractal compression was made using 32 · 32 pixels in domain blocks and 16 · 16 pixels in range blocks.*

figure 3.8. The use of luminance shift Δg_i shows resemblance to a normal distribution whereas the distribution of contrast scaling α_i has a clear overweight in the lower end. The peak at $\alpha_i = 6.4$ is a result of the limitation given by the six bit that are used to store this value. All values of α_i above 6.4 are assigned this maximum number. The use of isometries ι_i displays a fairly equal distribution between the different transformations, which is in opposition to [6], but what is to be expected from the procedure – no transformation should have a higher probability than any other.

To illustrate the distances from each range block to the corresponding domain block figure 3.9 shows the distance, distance Δ from the mid of a range block (x_r, y_r) to the mid of the matching domain block (x_d, y_d) measured in pixel values as

$$\Delta = \sqrt{(x_r - x_d)^2 + (y_r - y_d)^2} \tag{17}$$

The graph indicates that a range block is typically not located in near vicinity of its matching domain block. This result is contrary to the results presented in [6].

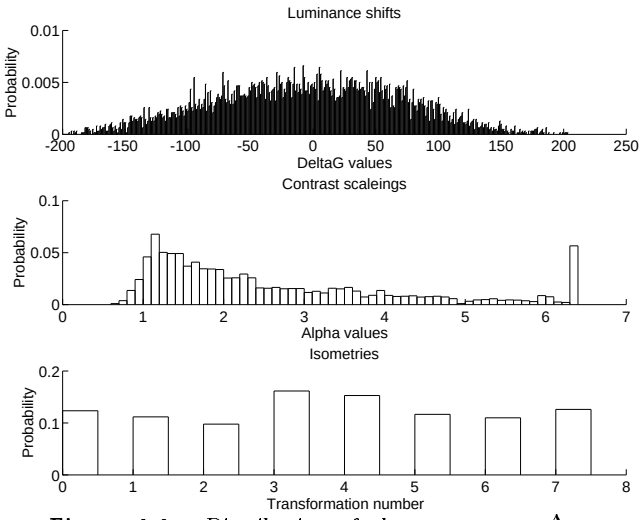


Figure 3.8: *Distribution of the parameters, Δg_i, α_i and Transformation number in the fractal code for 'Lenna'*

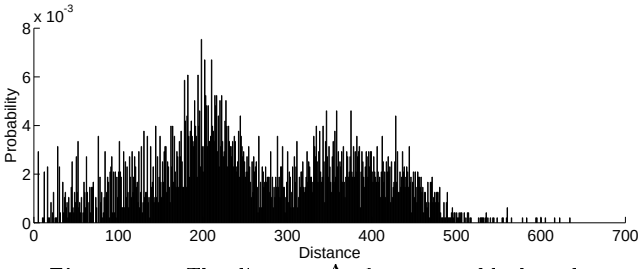


Figure 3.9: *The distance, Δ, from range block to domain block measured in pixels.*

Computing the bitrate

To store or transmit a fractal code in a file it is necessary to consider how the code is presented in the most compact but still fully reconstructable way. If we use a recursive approach, we can describe a fractal code in an n level quad-tree with use of a three-bit configuration field, $I_c = 3$, to describe the 12 possible configurations. For every main block in any level we therefore need a three-bit field to describe the descent into sub blocks. The information about block sizes is implicit hidden with this quad-tree partitioning whereas other partitioning methods such as triangular-partitioning [7] or HV-partitioning [2] requires an explicit storage of this information.

The description of the transformation for a block, whether it is a shade- or non-shade block, is given in table 3.1. The amount of bit needed to express a non-shade block in our system exceeds data listed by [6], but since the fractal code is entropy coded this does not degrade our system. If the number of bits used to describe the scaling factor α was reduced it would result in a lower SNR but the influence on the compression ratio of this field is far from important.

Table 3.1: *Information contents for the different block types.*

Block Type	Parameter	Bit
Shade	Header	2
	g_i	8
	Total I_s =	10
Non Shade	Header	2
	D_i	7 + 7
	α_i	6
	Δg_i	8 + 1
	t_{n_i}	3
	Total I_{ns} =	34

Table 3.2: *Shannon-Fano entropy coding of the fractal code for 'Lenna'.*

Field	Ratio	Stored
Grey level g_0 and Δg	0.89:1	
Contrast scaling α	1.05:1	$\surd$
Isometri ι	0.87:1	
XY-coordinates	1.14:1	$\surd$
Shade/Non Shade headers	1.55:1	$\surd$
Main/Sub configuration	1.21:1	$\surd$
Total	1.10:1	

To compute the resulting bitrate for a complete image, we have to find the total number of shade blocks N_s and non-shade blocks N_{ns}. Adding the number of main/sub configurations N_c to this, the total bitrate is

$$\frac{N_c I_c + N_s I_s + N_{ns} I_{ns}}{P} \tag{18}$$

where $P = x_{size} \times y_{size}$ is the number of pixels in the image.

Entropy coding of the fractal code

A particular run of the Shannon-Fano entropy coding on the fractal code for 'Lenna' is indicated in table 3.2. Here it is noticed that the encoding procedure is not able to reduce the grey levels (which is a combined absorption g_0 and luminance shift Δg), nor the isometries. This could be expected since the grey levels are distributed over a fairly large area with possibly very low probability for some grey levels. The isometries are distributed quite equally over the eight possible choices so no compression is to be expected. Only parameters where a reduction is obtained are stored, as indicated in the table.

4 Discussion

In this paper we have described the nature of a digital image compression system based on a fractal theory of iterated contractive image transformations, mainly extracted from [1], and suggested an approach to an improvement of the system.

The encoding is based on the assumption that the redundancy present in an image can be reduced by exploiting the self-transformability, on a square block basis. The output of the encoding process, the fractal code, is a set of contractive transformations which represents the image and contain the complete information needed to reconstruct an approximation of the image.

The fractal code for an image need not to include the blocks which were used to construct the code. An approximation of the original image will emerge upon iteration of the transformations on any initial image.

A summarize of the main results:

- The compression scheme proposed by [1] has been implemented, and improved by an enhanced classification routine as well as an unbounded quad-tree partitioning.
- The computational complexity of the encoding algorithm has in practice been reduced from approximately n^4 to approximately n^3, where n is the side length of the image.
- An additional entropy coding of the fractal code has been implemented, thereby reducing the redundancy of the fractal code.

Suggestions of further improvements:

- The grey level spectrum of a block and a window could be convoluted, thereby removing the current disadvantage of the algorithm which computes the dominant grey level parameter. The signal analysis of the convolution, however, becomes somewhat more complicated and hence requires more computing capability.
- The implementation of the encoding could be parallelized giving an even faster system, as long as the encoding of each range block is independent from one another.

We conclude that the performance of the system presented in this paper is comparable with standard image compression techniques, regarding signal-to-noise ratios and compression ratios, although the performance depends, to some extent, on the type of image to be compressed, i.e. not all images are compressed equally good. The applications of the compression scheme could be expanded to including compression of audio data and compression of image sequences, in which the self-transformability between two adjacent frames ought to be exploitable. We strongly suggest a further investigation into these possibilities.

References

[1] Jacquin A. E. *Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations.* IEEE Transactions on Image Processing 1992, Vol. 1 No. 1.

[2] Fisher Y., Jacobs E. W. and Boss R. D. *Fractal Image Compression Using Iterated Transforms.* Naval Ocean Systems Center, San Diego, CA 92152-5000, published in Storer J. A. *Image and Text Compression.* Kluwer Academic Publisher, 1992.

[3] Dunford N. and Schwartz J. T. *Linear Operators.* New York, Wiley 1966.

[4] Ramamurthi B. and Gersho A. *Classified Vector Quantization of Images.* IEEE Transactions on Communication 1986, Vol. 34.

[5] Shanmugan K. S. *Digital and Analog Communication Systems.* Joh Wiley & Sons, 1979.

[6] Jacquin A. E. *Fractal Image Coding: A Review.* Proceedings of the IEEE october 1993, Vol. 81 No. 10.

[7] Fisher Y. *Fractal Image Compression.* The San Diego Super Computer Center University of California, San Diego. Unpublished.

[8] Netravali A. N. and Haskell B. G. *Digital Pictures – Representation and Compression.* Plenum Press, 1991.