Technical Report

Dept. for Pattern Recognition and Image Processing
Institute for Automation
Technical University of Vienna
Treitlstr. 3/1832
A-1040 Vienna AUSTRIA
Phone: +43 (222) 58801-8161
Fax: +43 (222) 569697
E-mail: Franck.Davoine@imag.fr

PRIP-TR-25 November 9, 1993

Fractal Image Compression Based on Adaptive Tessellations[1]

Franck Davoine, [2]

Abstract

The principle of fractal image coding presented in this paper is based on the theory of L-IFS (Local Iterated Function Systems). The algorithm exploits the fact that a real-world image is formed approximately of transformed copies of parts of itself. Thus, the construction of fractal codes is directly made on partitions of the image support. It is based on piece-wise similarities between blocks of different sizes.
The paper starts with a regular block based approach first proposed by Jacquin A.E. [12]. To improve the algorithm, adaptive partitions are proposed with, in particular, the Delaunay triangulation. The results show an improvement in computing times, compression ratios and visual quality of reconstructed images.

[1]This work has partly been supported by the Commission of the European Communities under ERASMUS grant ICP 92-A-2007/11

[2]F. Davoine is with Equipe INFODIS, Laboratoire TIMC-IMAG, URA CNRS D 1618, BP 53, 38041 GRENOBLE Cedex 09, FRANCE

1 Introduction

Fractal image compression is a relatively recent technique since was first proposed by M.F. Barnsley in 1988. The main purpose of this method is to find resolution independant models of images. Many natural objects such as trees, clouds, mountains or leaves can be approximated by fractals. Barnsley [1] proposed the use of the theory of iterated transforms (IFS) to create fractal shapes. His best known result is the computation of a fern which truly resembles a natural one.
An IFS (Iterated Function System) is composed of contractive mappings performed on a complete metric space. The fractal object so generated needs only few coefficients to be coded. Thus, the use of IFS has been extended to compress natural images. The most important difference in this case is that a natural image is generally not self-similar (formed of copies of its self), and thus the method has to be adapted [2]. Jacquin A.E. [12] first proposed an automatic algorithm to code "real-world" images. The main idea is to exploit the image redundancy and to state that an image is formed of transformed copies of parts of itself. He showed that partitioning images into square blocks, and designing discrete transformations acting blockwise, approximates the original image by a self-similar one. He refered to this approach as fractal block coding. Many researchers subsequently compressed images with a similar approach [14, 3, 10, 6]. Others obtained fractal approximations of image blocks, based on IFS with probabilities [15]. Another possibility is to compute the parameters of the transformations directly from local invariant features of the image. This last idea has been succesfully implemented on a pixel row of an image [13].
In this paper, we propose a method similar to that of Jacquin. The originality of our approach is the use of Delaunay tessellation to partition the image support. The approach benefits from properties of triangulation, such as adaptivity, and non-rigidity. In **section 2** we present the theoretical foundations of this approach and introduce the L-IFS (Local-IFS) which makes it possible to automaticaly code natural images. The compression procedure is presented in **section 3** followed by the reconstruction of an approximation of the original image in **section 4**. Results are presented on a square regular partition in **section 5**. **Section 6** describes simple adaptive partitions of the image support. Then, we introduce Delaunay triangulation and explain how to obtain an adapted partition with a split and merge approach. This tessellation is compared with the regular square partition and the quadtree in order to encode an image. The encoding-decoding algorithm with triangles is explained in **section 7**. **Section 8** discusses compression aspects of the method. Lastly, results are presented in **section 9**.

2 Theoretical background

Let us consider a metric space (X, d) in which a digital image is a point. The metric d is a distance we will define below (section 3). A contractive transformation in this space is defined as:
$W : X \to X$

$\exists s < 1$ such that $\forall P, Q \in X$, $d(W(P), W(Q)) \leq s.d(P, Q)$
The real s in this case is defined as the contractivity factor of the operator W.
Such a contractive operator has an unique fixed point $\mathcal{A}_{\sqcup}$ in X, such that $\mathcal{A}_{\sqcup} = W(\mathcal{A}_{\sqcup})$. The iteration of the operator W converges to the point $\mathcal{A}_{\sqcup}$, starting with any point of X (Fig. 1). More precisely, we have $\lim_{n\to\infty} W^{o_n}(B) = \mathcal{A}_{\sqcup}$, $\forall B \in X$.
A central theorem (collage theorem), proposed by M.F. Barnsley, makes it possible to find an operator W whose fixed point is close to a given one. This inverse problem consists of seeing the image A to code as an approximation of the fixed point of the operator W. In this case, $A \simeq W(A)$. The operator W returns a result close to the original image A, starting with any image as explained above. The collage theorem states that : $d(A, \mathcal{A}_{\sqcup}) \leq \frac{1}{1-s} d(A, W(A))$, where $\mathcal{A}_{\sqcup}$ is the fixed point of W. If we can find the operator W which minimizes the distance between A and $W(A)$, the result of iterations of W on any initial set resembles the set A, providing that the contraction factor s not too close to 1. The main problem in coding a real-world image is the construction of the operator W in

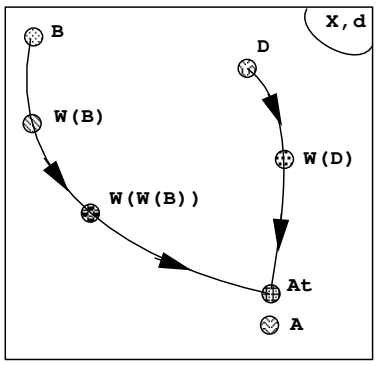


Figure 1: Convergence process.

such a way that $A \simeq W(A)$. A.E. Jacquin proposed a solution. He partitiones the image into non-overlapping blocks R_i and the operator W is composed of affine transforms acting on these. In this case, a L-IFS (Local Iterated Function System) is composed of a collection of local contractive transforms defined as $\omega_i(A \cap D_i)$, where $A \cap D_i$ is the restriction of A to the part D_i of the image A.
Thus, the operator W is defined as $W(A) = \bigcup_{i=1}^{N} \omega_i(A \cap D_i)$, and returns the union of transformed parts of the image A. It has been shown that it is not necessary that each of the transforms ω_i have a contractivity factor smaller than one. The necessary contractivity requirement is that W be eventually contractive [11]. This means that the operator W must be composed of sufficiently contractive transforms ω_i with respect to the expensive ones in order to have the m^{th} iterate W^{o_m} contractive ($m \in \mathbb{N}$).
The collage theorem states that if it is possible to cover the image A by transformed parts of itself (with contraction mappings ω_i) so that the result is close to the image A, then

the collection of transforms ω_i approximately defines the image A. The collage not being exact, we have an approximation of the original image during the decoding phase.

3 Overview of the encoding principle

Let us consider a grey level image A to be encoded. We see it as the attractor of the L-IFS we want to find. Thus $A = W(A) = \bigcup_{i=1}^{N} \omega_i(A \cap D_i)$. The transform W has to be eventually contractive, under an appropriate metric.
The algorithm consists in using a two-level partition of the image support. One of the two levels returns blocks D_i (partition D), and the other returns blocks R_i of a smaller or equal area (partition R). The partition R generates non-overlapping blocks. More details will be given on their construction. The encoding algorithm finds, for each block R_i, a transformed domain block D_i which is very close to R_i. The block D_i can be found anywhere in the partition D. Then, if we perform the collage of the transformed blocks D_i on the blocks R_i, we have $W(A)$ nearly equal to A. In this case, the collage theorem is satisfied and the operator W encodes the image (Fig. 2).

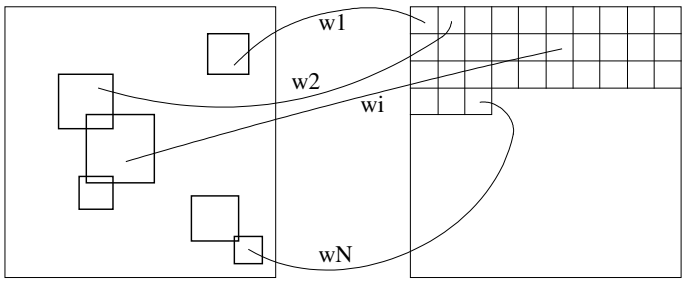


Figure 2: The encoding principle of an image A. Left: partition D. Right: partition R, on which the collages are done. The transformed image $W(A)$ (right) has to be close to the image A (left) in order to respect the collage theorem.

The mappings ω_i used in our implementation can be written as:

$$\omega_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}.$$

$\omega_i(A \cap D_i) = t_i(v_i(A \cap D_i))$ is composed of two transformations. One geometric transform acting in the plane (transf. v_i, param. $a_i, b_i, c_i, d_i, e_i, f_i$) and a grey level one modifying the pixels intensity z (transf. t_i, param. s_i, o_i). s_i, o_i control the contrast and the brightness

of the pixels grey level respectively.
The similarity (in terms of grey level) between the block R_i and the transformed block D_i is measured with the square error (SE). It is a classical metric to measure the distance between two images. The square error is given by: $d(A \cap R_i, \omega_i(A \cap D_i)) = \sum_{n=1}^{n_0}(s_i.d_n + o_i - r_n)^2$ where d_n and r_n are respectively the intensities of the pixels of the blocks D_i and R_i, and n_0 the number of pixels included in $A \cap R_i$. s_i, o_i are the coefficients of ω_i.
The contractivity of the transform W is thus controled by the parameters s_i. These must have to be less than 1, in order to insure eventual contractivity. In the general case, the collages are made from blocks D_i to blocks R_i of smaller base area, but this is not a required condition. What is important is contractivity of the grey level transformation [4].

4 Decoding from fractal code

The fractal code for the compressed image A is composed of a collection of N contractive transforms ω_i(x,y,z). The decoding consists in iterating the operator W, starting with any initial image B. The reconstructed image is given by $A \simeq \lim_{n\to\infty} W^{o_n}(B)$. One iteration consists in scanning the blocks R_i (in the same order as during the coding step) and in applying the affine transformation ω_i on their corresponding domain block D_i (Fig. 3). We usually need 8 to 12 iterations to converge to the original image. The number of iterations depends on the resolution of the image.

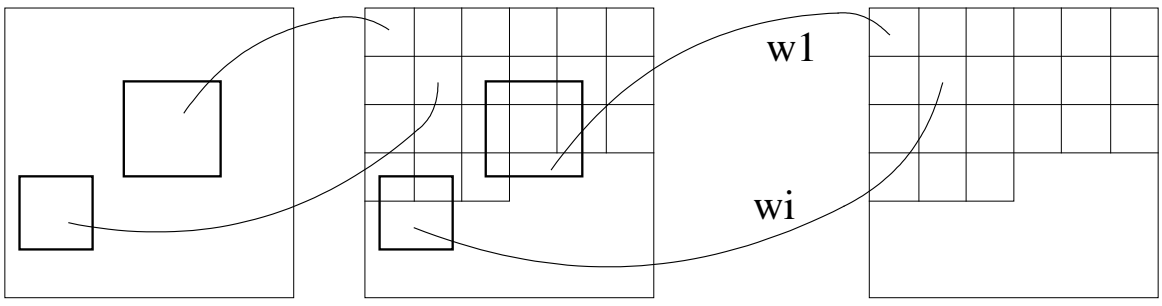


Figure 3: The decoding principle from a fractal code. Left to right: any initial image, first iteration and last iteration (the result converges to the original image).

5 Results on a regular squares partition

In this section we present results with a simple algorithm. The image is partitionned into non-overlapping regular squares of sizes $x_R * x_R$ and $2x_R * 2x_R$. The small squares compose

the partition R, the others the partition D.
The decoding steps are shown in Fig. 4. In Fig. 4.a, 1024 transformations are required. The compression ratio calculation is similar to that explained in section 8. Partition D is composed of 256 squares which can be coded in a list.

(a) 1024 non-overlapping squares of size $8 * 8$ in partition R. Compression ratio = 20.4:1, PSNR = 27.2 dB.

(b) 4096 non-overlapping squares of size $4*4$ in partition R. Compression ratio = 5.1:1, PSNR = 32 dB.

Figure 4: Two decompressions of the image "femme" of size $256*256$. From top to bottom, left to right: original image (A), first iteration ($W(B)$), second ($W^{o_2}(B)$), third ..., 8th iteration (decoded image) and enhanced error image by a factor of 4. B is a white image.

Remarks can be made on those two results:
If the squares in partition R are too large, details in the reconstructed image are not visible.

If a square covers a small detail, it will not find a corresponding domain square including the same detail. In that case, the error between the blocks is important, it does not allow verification of the collage theorem.
If the squares are too small, the compression ratio is also small, due to the large number of affine transforms to code.

6 Adaptive partitions of the image support

To construct the fractal code, we need to partition the image support. Different partitions have been proposed, using regular squares, quadtrees, rectangles and triangles. The main point of this chapter will be concerned by Delaunay triangulation.

6.1 Quadtree

The Quadtree is defined as a tree of degree 4 [17]. The root of the tree is the entire image. If the image has a constant grey level, the root node is labeled with this value. Otherwise, four descendants are added to the node. The process is then repeated recursively for each

Figure 5: Quadtree evolution. The final Quadtree has 5719 squares.

of those nodes. If a block has a constant value, its node is a leaf node of the tree. An example of an image with its corresponding quadtree is shown in Fig. 5. The split criterion can also be based on variance, or gradient inside the blocks.
The main disadvantages of this scheme are that it returns too many squares if the size of the blocks is small. Moreover, it is shift-variant. Two images that differ only by a translation may have two very different quadtrees.

6.2 Partitions H-V

The partition H-V for fractal block-coding has been proposed by Y. Fisher [10]. The image is partitioned into two rectangular regions, and then, each rectangle is recursively partitioned to form two new rectangles. The process stops when a given criterion is satisfied. The split process can be done in order to obtain a maximum of rectangles with diagonally oriented edges in their interior, and thus a maximum of similarities between the blocks. The advantage of this scheme is that the position of rectangles is variable, but the disadvantage is that their orientation is limited to 90 degree angles. The horizontal and vertical edges in the image are well covered but the other oriented shapes require many rectangles.

6.3 Delaunay triangulation

Delaunay triangulation offers good properties of regularity. The unconstrained orientation of triangles makes it possible to have a data dependant partition [16].

6.3.1 Definitions

Let us consider a finite set of points $T = \{t_n\}$. It has been demonstrated that the Delaunay graph is the unique triangulation with "empty circles" (Fig. 6). That is: the circumcircle of every triangle (t_i, t_j, t_k) of the Delaunay triangulation does not contain any other points t_l of T in its area. The dual diagram is called the Voronoi tessellation. It provides a

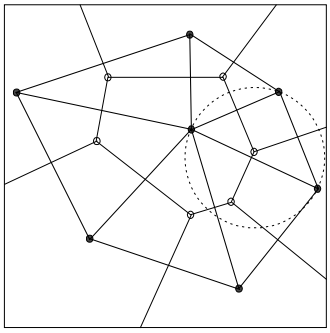
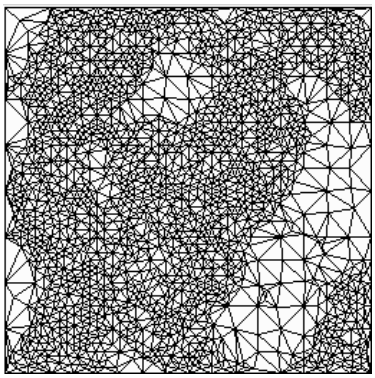


Figure 6: Delaunay triangles and Voronoi polygons. The points t_n are in black. The white points are the centers of the circumcircles.

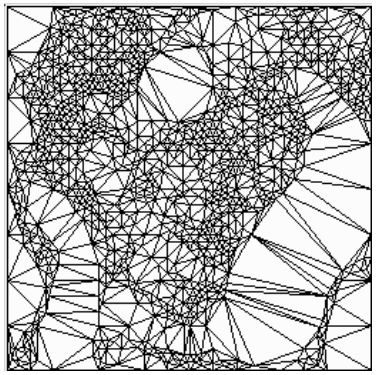
partition of the space into Voronoi polygons. Given the same set of points $\{t_n\}$ called seeds, for every point t_i in T, the polygon associated with t_i is the part of the space where every point in its interior is closer to t_i than to any other point of T.

6.3.2 Split and Merge algorithm based on Delaunay triangulation

(a) Split steps initialized on a white image.



(b) Split: 3476 triangles



(c) Split and Merge: 2157 triangles

Figure 7: Delaunay tessellation of the image "femme".

Different algorithms have been proposed to construct the Delaunay triangulation.
One advantage of our method (an incremental approach) is that the image orients the evolution and location of the polygons. Essentially there must be a high density of seeds (triangles) in regions including details in the image. This principle can be understood in terms of variance, or gradients computed on pixels belonging to the interior of triangles [8, 5, 7, 9].
In terms of data structure, a graph environment is used in order to facilitate the manipulation of the triangles.
The algorithm works by local modifications of the triangulation when a new point is inserted. It proceeds in two steps which are called Split and Merge.
We start with a small number of points regularly distributed on the image support. The split step (Fig. 7.a-b) consists in adding a point on the barycenter of each non homogeneous triangle (variance or gradient criteria). We continue the split process until convergence. Thus we stop when either the triangles are homogeneous or the surfaces of the triangles are less than given threshold. The merge step (Fig. 7.c) consists in deleting neighboring triangles having similar mean grey levels. Those triangles are suppressed from the graph. We note that the number of triangles is inferior to the number of squares obtained in previous algorithms.

7 Encoding-Decoding algorithm with triangles

In our implementation, the blocks R_i and D_i arise from Delaunay triangulations.
The partitions are performed in a split and merge approach initialised on a regular distribution of points.
The second level (returning the blocks D_i) does not need to have small triangles, and we stop the split process before returning the blocks R_i.
We finally obtain a domain block partition with large triangles and a range block partition well adapted to the image information, with smaller triangles. Moreover, the partition R is denser in the detailed parts of the image.
The coding process based on those triangulations consists in selecting, for a given block R_i, a block D_i in the partition D which gives the smallest distorsion (distance SE) between $A \cap R_i$ and $\omega_i(A \cap D_i)$. The search is done using a walk in the Delaunay graph of the partition D. The size of the triangle R_i is selected to be smaller or equal to the size of the triangle D_i. We recall that there are 6 different ways to map one triangle onto another, therefore we try 6 possibilities for each mapping of a block D_i onto a block R_i.
Furthermore, the blocks D_i have more pixels in their area than the blocks R_i. The Square Error is computed between all the pixels in R_i and their images in D_i. Blocks D_i are thus sub-sampled.
The coding algorithm can be improved by a classification of the triangles. This speeds up the matching process and does not allow range blocks to be in correspondance with very different domain blocks. These mistakes are possible with the least-squares calculation. The classification creates different classes of triangles. In this case, the use of a contour

image allows separation of the triangles in two classes: edge triangles and others. Thus, the search, for a given edge range blocks, will be done through a list of edge domain blocks. Of course, the classification algorithm must be robust, so that new errors in the matching are not introduced.

8 Compression Ratio

To encode an image we need to store the coefficients of the N mappings ω_i composing the L-IFS. One mapping is needed for each block R_i in the partition R. The Delaunay tessellation is coded in a graph environment. Thus, for one transformation ω_i, it is only necessary to code:

- the position of the triangle D_i in Delaunay graph
- the orientation for the collage (6 possibilities to map a triangle onto another)
- the scale coefficient (s_i)
- the offset coefficient (o_i)

It has been verified [11] that 5 bits to code the coefficient s_i and 7 bits for o_i are sufficient to provide a good quality reconstructed image. The positions of the triangles D_i in the graph depend on N. The orientation needs 3 bits to be coded.
In addition, we must code the way to obtain the image adapted partitions R and D. Since we always start with a regular distribution of points to construct the Delaunay triangulation, it is only usefull to code the split and merge process, with 1 bit per split and 1 bit per merge. The division of a triangle during the split steps is coded with a 1, the non-division with a 0. The obtained string of binary codes can also be compressed with classical techniques. For example, the partition in Fig. 7c. is coded by an uncompressed string of 9841 bits.
The total number of bits necessary to code the L-IFS is thus given by: $N_{tot} = N(15+M)+X$ with N = total number of blocks R_i, X = number of bits to code the partition processes and $M = E[\log_2 (\text{nb. of blocks } D_i + 1)]$ = number of bits to code the position of the block D_i in the graph (E means the integer part). If the number of blocks D_i is less than 1024, $M = 10$.
For a grey scale image of size 256*256, 8 bpp, the compression ratio is given by: $T_c = \frac{256*256*8}{N_{tot}}$.

9 Results and discussion

We have presented results on a 256*256, 8 bits/pixel images. The formula to calculate the peak-signal-to-noise ratio is given by : $PSNR = 10 * log_{10}(\frac{(2^n-1)^2}{\frac{1}{n_0}\sum_{i=1}^{n_0}(\mu_i-\nu_i)^2})$ with n the number of bits per pixel in the original image, n_0 the number of pixels in the image, μ_i and ν_i the intensities of pixels in the original and reconstructed image respectively.
First we constructed the fractal codes of the image "femme" on a Delaunay triangulation,

using the split and merge approach (Fig. 8.a). The resulting tessellation returns irregular triangles which are image constrained. Large triangles appear in homogeneous parts of the image. There are 2157 triangles in the partition R, and 1013 in the partition D. The search for the matching of range blocks is done among a limited number of triangles D_i and not anywhere in the image. The method is therefore computationaly efficient when compared to the one using square partitions. (compression time = 15 minutes, decoding = 5 secondes, on a Silicon Graphics Indigo^{TM} R4000). No classification was made.

(a) Partition R of figure 7.c : Compression ratio = 9.5:1, PSNR = 30 dB.

(b) Compression ratio = 7:1, PSNR = 30 dB.

Figure 8: Decompressions of images of size $256 * 256$. From top to bottom, left to right: original image, first, seconde, third ..., 10th iteration (decoded image) and enhanced error image by a factor 4. The iterations start with a white image.

Of course, in order to preserve a good fidelity of reconstructed image with respect to the original, the two partitions have to present a maximum number of similar blocks, and also to be image dependant. The decoded image does not present "block effects" as in the situation where the methods based on square partitions are used. The unconstrained orientation of the triangles gives good results. Large triangles appear in homogeneous parts of the image. The number of triangles in the partition R is also less important (2157 triangles). The decoded image of LENA is shown in Fig. 8.b.

Acknowledgments

I wish to thank Etienne Bertin for his cooperation in the adaptive partition programs and am grateful to Jean-Marc Chassery and again to Etienne Bertin for their enlightening discussions during this work.

References

[1] M.F. Barnsley. *Fractal everywhere.* Academic Press, New York, 1988.

[2] M.F. Barnsley and L.P. Hurd. *Fractal Image Compression.* AK Peters Ltd., Wellesley, 1993.

[3] J.M. Beaumont. Image data compression using fractal techniques. *BT TechNol. J.*, 9(4):93–109, 1991.

[4] T.J. Bedford, F.M. Dekking, M. Breeuwer, M.S. Keane, and D. Van Schoonneveld. Fractal coding of monochrome images. Technical Report 92-99, Faculty of Technical Mathematics and Informatics, Delft, The Netherlands, 1992.

[5] E. Bertin, F. Parazza, and J.-M. Chassery. Segmentation and measurement based on 3D voronoi diagram: Application to confocal microscopy. *Computerized Medical Imaging and Graphics*, 17:0–8, 1993.

[6] A. Bogdan and H.E. Meadows. Kohonen neural network for image coding based on iteration transformation theory. *SPIE Neural and Stochastic Methods in Image and Signal Processing*, 1766:425–436, 1992.

[7] A. Bowyer. Computing dirichlet tessellations. *The computer J.*, 24(2):162–166, 1981.

[8] J.-M. Chassery and M. Melkemi. Diagramme de Voronoï appliqué à la segmentation d'images et à la détection d'événements en imagerie multi-sources. *Traitement du Signal*, 8(3):155–164, 1991.

[9] X. Chen and F. Schmitt. Split-and-merge image segmentation based on Delaunay triangulation. In *Proc. of The 7th Scandinavian Conf. on Image Analysis, Aalborg, Denmark*, pages 910–917, aug. 13-16 1991.

[10] Y. Fisher. A discussion of fractal image compression. In *Chaos and Fractals, New Frontiers of Science*, pages 903–919. H.O. Peitgen, H. Jürgens, D. Saupe, Springer-Verlag, New-York, 1992.

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

[12] A.E. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, 1992.

[13] W.G. Kropatsch, M.A. Neuhausser, and I.J. Leitgeb. Iterated functions systems – a direct discrete approach with pyramids. In *Proc. of the 16th OAGM - Meeting Vienna*, May 6-8 1992.

[14] S. Lepsoy, G.E. Oien, and T.A. Ramstad. An inner product space approach to image coding by contractive transformations. *ICASSP*, 3:2773–2776, 1991.

[15] D.M. Monro and F. Dudbridge. Fractal approximation of image blocks. *ICASSP*, 3:485–488, 1992.

[16] J.P. Preparata and M.I.S. Shamos. *Computational Geometry, an Introduction*. Springer Verlag, NewYork, 1988.

[17] H. Samet. Region representation: quadtrees from binary arrays. *CVGIP*, 13:88–93, 1980.