Fractal Image Compression

Miroslav Galabov

Abstract: *The paper gives and introduction on Image Coding based on Fractals and develops a simple algorithm to be used as a reference design.*

Key words: *Image Coding, Fractal Theory, Image Compression, Iterated Function System.*

Introduction

With the advance of the information age the need for mass information storage and fast communication links grows. Storing images in less memory leads to a direct reduction in storage cost and faster data transmissions.

Images are stored on computers as collections of bits representing pixels or points forming the picture elements. Since the human eye can process large amounts of information (some 8 million bits), many pixels are required to store moderate quality images.

Most data contains some amount of redundancy, which can sometimes be removed for storage and replaced for recovery, but this redundancy does not lead to high compression ratios.

The standard methods of image compression come in several varieties. The current most popular method relies on eliminating high frequency components of the signal by storing only the low frequency components (Discrete Cosine Transform Algorithm).

This method is used on JPEG (still images), MPEG (motion video images), and H.261 (Video Telephony on ISDN lines) compression algorithms.

What is Fractal Image Compression?

Imagine a special type of photocopying machine that reduces the image to be copied by half and reproduces it three times on the copy [1]. What happens when we feed the output of this machine back as input? Figure 1 shows several iterations of this process on several input images. We can observe that all the copies seem to converge to the same final image. Since the copying machine reduces the input image, any initial image placed on the copying machine will be reduced to a point as we repeatedly run the machine; in fact, it is only the position and the orientation of the copies that determines what the final image looks like.

The way the input image is transformed determines the final result when running the copy machine in a feedback loop. However we must constrain these transformations, with the limitation that the transformations must be contractive (see contractive box), that is, a given transformation applied to any two points in the input image must bring them closer in the copy. This technical condition is quite logical, since if points in the copy were spread out the final image would have to be of infinite size. Except for this condition the transformation can have any form. In practice, choosing transformations of the form

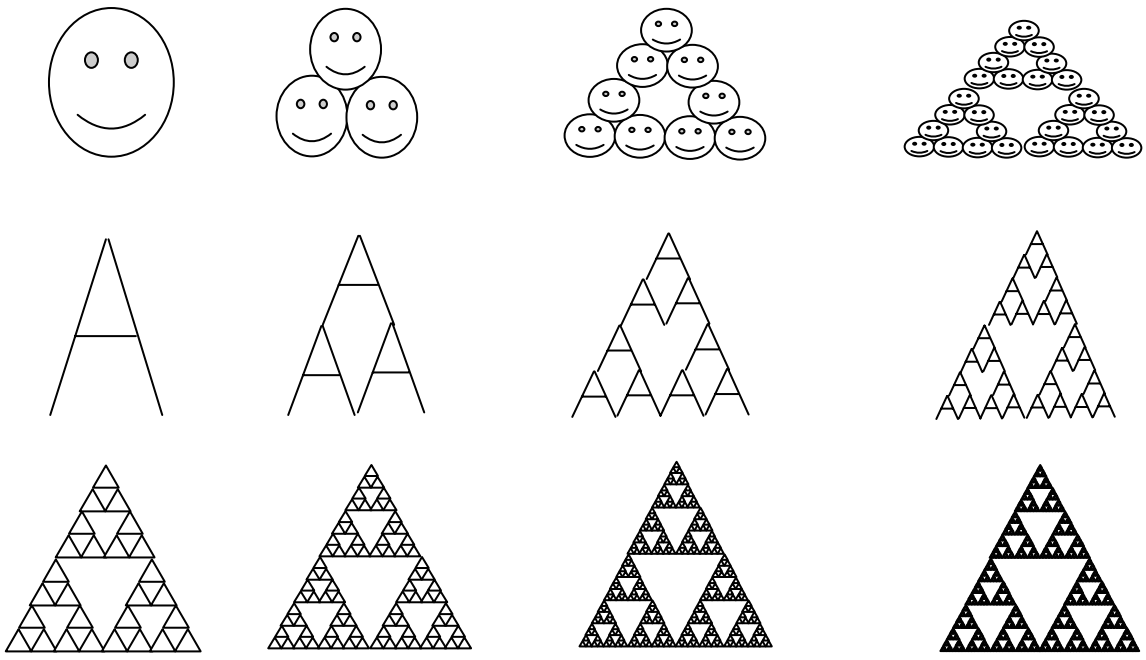
$$w_i \begin{vmatrix} x \\ y \end{vmatrix} = \begin{vmatrix} a_i b_i \\ c_i d_i \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix} + \begin{vmatrix} e_i \\ f_i \end{vmatrix},$$

where x,y – coordinates;

a,b,c,d,e,f – coeficients,

is sufficient to generate interesting transformations called *affine transformations* of the plane. Each can skew, stretch, rotate, scale and translate an input image.

A common feature of these transformations that run in a loop back mode is that for a given initial image each image is formed from a transformed (and reduced) copies of itself, and hence it must have detail at every scale. That is, the images are *fractals* [2].



Initial Image First Copy Second Copy Third Copy

Figure 1. The first three copies of the Initial Image

In [2], it was suggested that perhaps storing images as collections of transformations could lead to image compression. His argument went as follows: the image in Figure 1 looks complicated yet it is generated from only 4 affine transformations.
Each transformation w_i is defined by 6 numbers a_i, b_i, c_i, d_i ,e_i and f_i, which do not require much memory to store on a computer (4 transformations x 6 numbers transformations x 32 bits /number = 768 bits). Storing the image as collection of pixels, however required much more memory. So if we wish to store a picture of a fern, then we can do it by storing the numbers that define the affine transformations and simply generate the fern whenever we want to see it. Now suppose that we were given any arbitrary image, say a face. If a small number of affine transformations could generate that face, then it too could be stored compactly. The trick is finding those numbers.

Contractive Transformations

A transformation w is said to be contractive if for any two points P1, P2, the distance

$$d(w(P1), w(P2)) < sd(P1, P2),$$

for some *s* < 1, where d = distance. This formula says the application of a contractive map always brings points closer together.

The Contractive Mapping Fixed Point Theorem

This theorem says something that is intuitively obvious: if a transformation is contractive then when applied repeatedly starting with any initial point, we converge to a unique fixed point. If X is a complete metric space and W:X->X is contractive, then W has a unique fixed point $|W|$.

This simple looking theorem tells us how we can expect a collection of transformations to define an image.

The image compression scheme describe later can be said to be fractal in several senses [3]. The scheme will encode an image as a collection of transforms that are very similar to the copy machine metaphor. Just as the fern has detail at every scale, so does the image reconstructed from the transforms. The decoded image has no natural size, it can be decoded at any size. The extra detail needed for decoding at larger sizes is generated automatically by the encoding transforms. One may wonder if this detail is "real"; we could decode an image of a person increasing the size with each iteration, and eventually see skin cells or perhaps atoms. The answer is, of course, no. The detail is not at all related to the actual detail present when the image was digitised; it is just the product of the encoding transforms which originally only encoded the large scale features. However, in some cases the detail is realistic at low magnifications.

The compression ratio for the fravtal scheme is hard to measure since the image can be decoded at any scale [4].

Encoding Images

The previous theorem tells us that transformation W will have a unique fixed point in the space of all images. That is, whatever image (or set) we start with, we can repeatedly apply W it and we will converge to a fixed image.

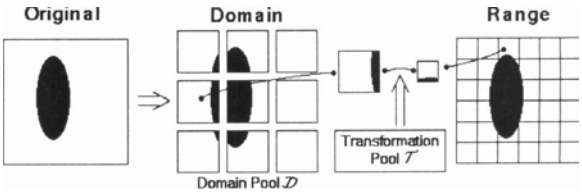


Figure 2. The encoding process

Suppose we are given an image f, that we wish to encode. This means we want to find a collection of transformations w_1, w_2, …w_N and want f to be the fixed point of the map W. In other words, we want to partition f into pieces to which we apply the transformations w_i and get back the original image f. Typical image of a face, does not contain the type of self-similarity. The image does contain other type of self-similarity. These transformed parts do not fit together, in general, to form an exact copy of the original image, and so we must allow some error in our representation of an image as a set of transformations [5,6].

Proposed Algorithm

The following example suggests how the Fractal Encoding can be done.

Suppose that we are dealing with a 128 x 128 image in which each pixel can be one of 256 levels of grey. We called this picture Range Image. We then reduce by averaging (down sampling and lowpass-filtering) the original image to 64 x 64. We called this new image Domain Image. We then partitioned both images into blocks 4x4 pixels (see Figure 3). We performed the following affine transformation to each block:

$$(D_{i,j}) = \alpha D_{i,j} + t_0 ,$$

where α - contrast scaling($\alpha = [0,1], \alpha \in R$) и t_0 - luminance shift ($t_0 \in [-255,255], t_0 \in Z$).

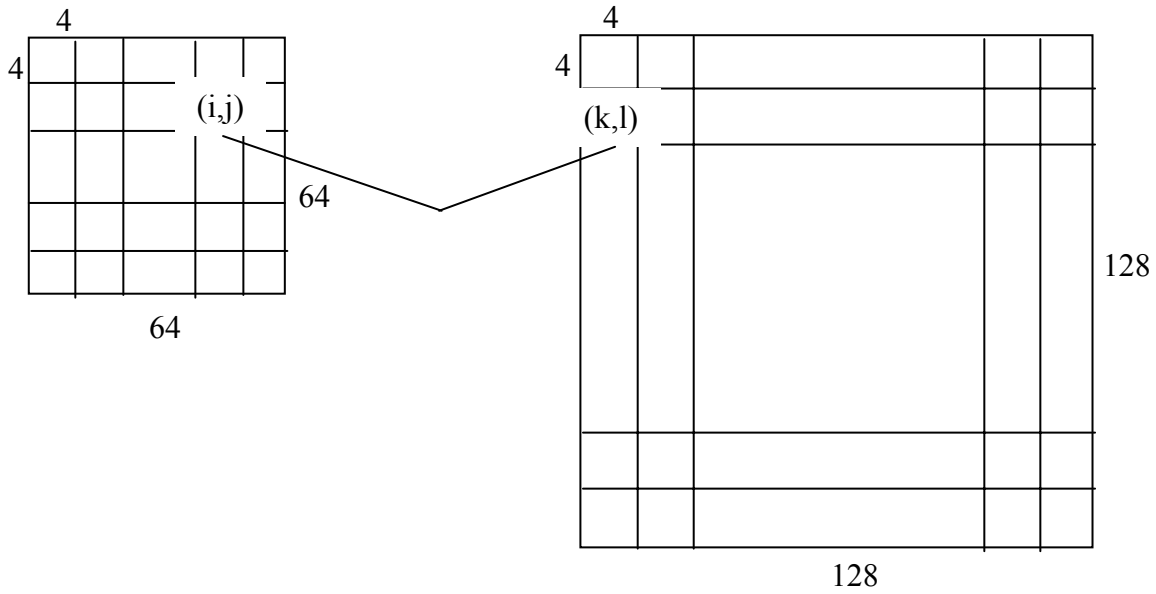
In this case we are trying to find linear transformations of our Domain Block to arrive to the best approximation of a given Range Block. Each Domain Block is transformed and then compared to each Range Block $R_{k,l}$. The exact transformation on each domain block, i.e. the determination of α and t_0 found minimizing

$$\min \sum_{m,n} (R_{k,l})_{m,n} - (T(D_{i,j}))_{m,n} ,$$

where m, n, = 2 or 4 (size of blocks).

Each transformed domain block $T(D_{i,j})$ is compared to each range block $R_{k,l}$, in order to find the closest domain block to each range block. This comparison is performed using the following distortion measure:

$$d_{l_2}(T(D_{i,j}), R_{k,l}) = \sum_{m,n} (T(D_{i,j}) - (R_{k,l})_{m,n})^2 .$$



D – domain R – range

Figure 2. Partition of Range and Domain

Each distortion is stored and the minimum is chosen. The transformed domain block which is found to be the best approximation for the current range block is assigned to that range block, i.e. the coordinates of the domain block along with its α and t_0 are saved into the file describing the transformation. This is what is called the *Fractal Code Book.*

$$T(\alpha, t_0, (i,j))_{best} \Rightarrow R_{k,l}$$

The reconstruction process of the original image consists on the applications of the transformations describe in the fractal codebook iteratively to some initial image Ω_{init} until the encoded image is retrieved back. The transformation over the whole initial image can be describe as follows:

$$\Omega 1 = \eta(\Omega_{init})$$
$$\Omega 2 = \eta(\Omega_1)$$
$$\Omega 3 = \eta(\Omega_2)$$
$$.....$$
$$\Omega n = \eta(\Omega_{n-1})$$

η can be expressed as two distinct transformations:

$$\eta = T(\Omega)\Psi(\Omega),$$

where $T(\Omega)$ represents the down sampling and lowpass filtering of an image 0 to create a domain image e.g. reducing a 128x128 image to a 64x64 image as we describe previously.
$\Psi(\Omega)$ represents the ensemble of the transformations defined by our mappings from the domain blocks in the domain image to the range blocks in the range image as recorded in the fractal. Ωn will converge to a good approximation of $\Omega orig$ in less than 7 iterations.

a)original image b)decoded by using 2x2 square c) decoded by using 4x4 square

Figure 3.Results of decoding after 6 iterations

Results

We have decoded Lena (128x128) by using the algorithm described above and block scheme on figure 4b as the process has started with a black image. This is performed using the 2x2, and 4x4 blocks. The results are given in Table 1 and can be observed on Figure 3b and c.

Table 1

Parametars	**Method 2x2**	**Method 4x4**
Blosk size	2x2	4x4
Number of iterations	6	6
Time to encode	10 min.	55 s.
Time to decode	45 s.	28 s.
Size of the code book	16 384 bytes	6 144 bytes
Sample to noice ratio	27 dB	21 dB
Mean pixel difference between original and decoded image	115	95

From the results presented in Table 1 we can make the conclusion that when encoding by small range blocks (<8) we achieve a high SNR but a low compression ratio. With higher blocks, the compression is increased but the details in the image are lost.

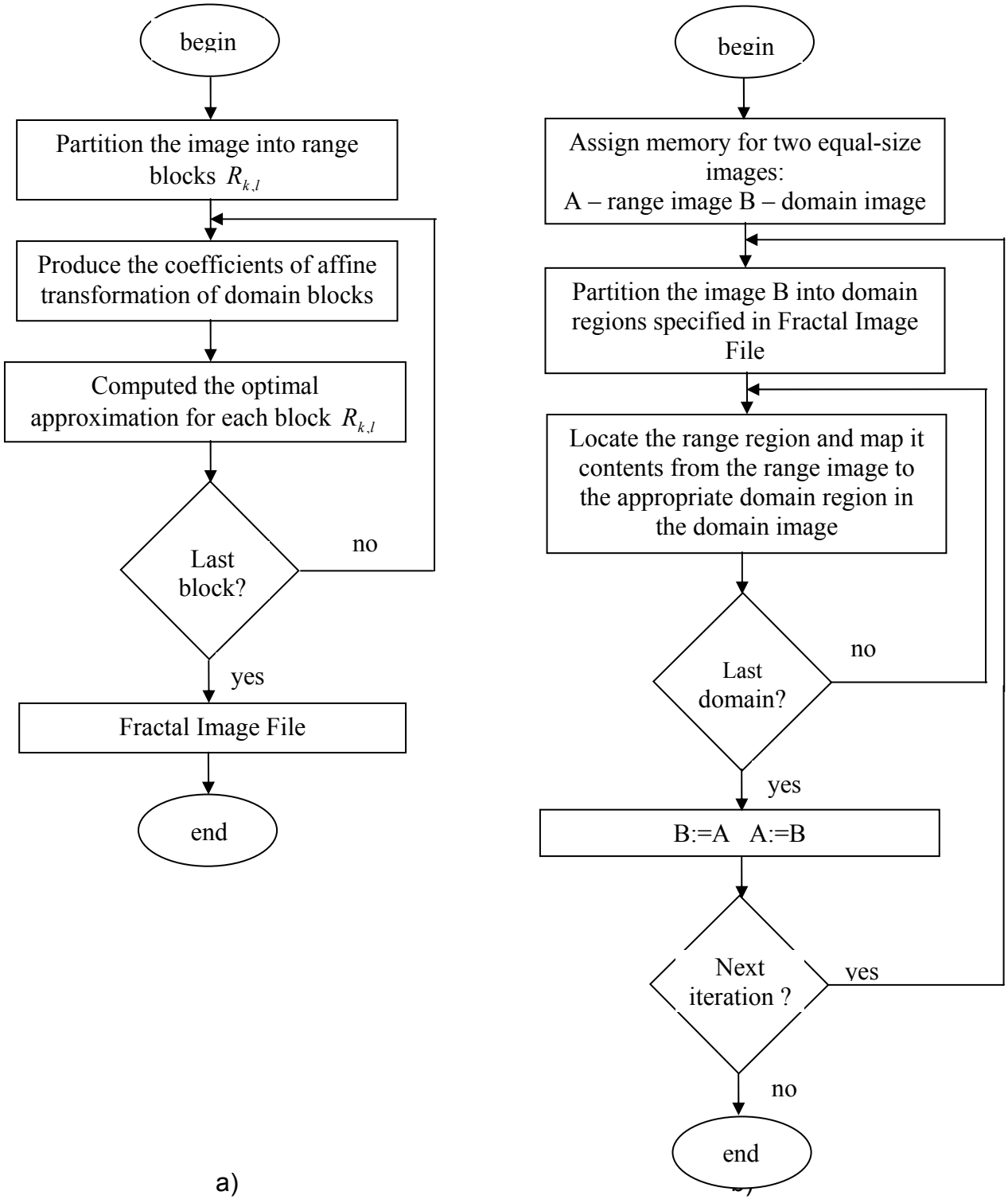


Figure 4. Block shemes of the proposed encoder (a) and decoder (b)

Conclusion

The results presented above were obtained using the MATLAB Software. A great improvement on the encoding/decoding time can be achieved with the use of real Digital Signal Processor, for example TMS320C50 of Texas Instruments.

A weakness of the proposed reference design is the used of fixed size blocks for the range and domain images. There are regions in images that are more difficult to code than

others. Therefore, there should be a mechanism to adapt the block size ($R_{k,l}$,$D_{i,j}$) depending on the error introduced when coding the block.

The most important feature of Fractal Decoding is the high image quality when Zooming IN/OUT on the decoded picture. This type of compression can be applied in Medical Imaging, where doctors need to focus on image details, and in Surveillance Systems, when trying to get a clear picture of the intruder.

References

[1]Fisher,Y.Fractal Image Compression:Theory and Application.Springer Verlag, N.Y., 1995.

[2]Barnsley. M.Fractals Everywhere.Academic press.San Diego,1989.

[3]Peitgten,H.O.,D.Daupe, H.Jurgens. Fractals For Class Room.Springer Verlag,N.Y., 1991.

[4]Walach, E.,E.Karnin.A Fractal Based Approach to Image Compression. Proceedings of ICASSP, Tokio, 1986.

[5]Fisher,Y.,E.W.Jacobs,R.D.Boss.Fractal Image Compression Using Iterated Transforms. Kluwer Academic Publishers, Norwall, 1989.

[6]Jacquin, A.Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding. Doctoral Thesis, Georgia Institute of Technology,1989.

ABOUT THE AUTHOR

Assistant Prof. Miroslav Galabov, PhD, Department of Computer Systems and Technologies, University of Veliko Turnovo, Phone: +359 62 322 34, E-mail: plam@vali.bg