Vasileios Drakopoulos *

# FRACTAL-BASED IMAGE ENCODING AND COMPRESSION TECHNIQUES

*In computer science and information theory,* data compression*,* source coding*, or* bit-rate reduction *is the process of encoding digital information using fewer bits than the original representation. Specifically,* digital-image compression *is important due to the high storage and transmission requirements. Various compression methods have been proposed using different techniques to achieve high compression ratios.* Fractal image encoding *is a technique based on the representation of an image by contractive transformations.* Fractal-based image compression *methods belong to different categories according to the different theories they are based on. In this article, first we try to clarify the terminology used and then to comprehensively unveil the mathematical principle behind* fractal image compression *as well as to briefly overview a variety of schemes that have been investigated.*

***Keywords:*** *Approximation, coding, dimension, fractal, image compression, image encoding, interpolation, iterated function system, local, partitioned, recurrent, transformation.*

## 1. Introduction

The *fractal transform* was discovered in 1988 by Michael F. Barnsley. It is the basis for an image compression scheme which was initially developed by Alan D. Sloan and M. F. Barnsley at Iterated Systems, Inc. They used the similarities on different scales throughout images to assist in compression. Since then, the research literature on *attractor image coding, fractal-based image coding, fractal image coding, fractal-based image compression, fractal image compression, fractal-based image encoding, fractal image encoding* and *fractal image approximation* has experienced a rapid growth. First of all, let's understand the interconnections between these topics by clarifying the terms coding and encoding.

### 1.1 Etymology and Meaning

The origin of the word *code* is traced to the Latin word *codex*, a manuscript (hand-written) volume, which in turn comes from the Latin *caudex*, trunk of a tree or block of wood. It can be used as either a noun or a verb. When used as a noun, it means a collection of laws or a system of principles and rules (canons). In our context, it means a system of symbols and rules used as instructions (statements) to a computer or the symbolic arrangement of them – a computer program; for instance, instructions written by a programmer in a programming language are often called *source code*.

When *code* is used as a verb, it is commonly confused with *encode*. There is a strong difference between the two words:

- to *code* means to express information through a proper standard representation. In the context of computer science it refers to write or revise a computer program.
- to *encode* means to perform an operation that transforms some information from one representation form to another. Something that is performing the act of encoding is, for example, an encrypter or a compressor.

Saying 'image-coding schemes' or 'image-coding methods' literally means that the scheme or the method expresses information relative to an image. Saying instead 'image-encoding schemes' or 'image-encoding methods' literally means that the scheme or the method is an entity performing encoding on the images, that is, transforming an image from one representation form to another, which is the correct for our purpose. It looks like many computer scientists use this wrong wording as standard nomenclature.

### 1.2 Key Terms

*Compression* is commonly confused with the notion of encoding. Compression is done solely to lessen the number of symbols to represent given piece of information. It is achieved with the help of specific encoding of information. The process of reducing the size of a data file is popularly referred to as *data compression*,

* **Vasileios Drakopoulos**
School of Science & Technology, Hellenic Open University, Greece, E-mail: vasilios@tutors.eap.gr

although its formal name is *source coding* (coding done at the source of the data before it is stored or transmitted). *Image compression* is the application of data compression on digital images. The objective of image compression is to reduce irrelevance and redundancy of the image data in order to be able to store or transmit data in an efficient form. Compression can be either lossless or lossy.

The term *fractal* is coined by Benoît B. Mandelbrot in 1975, about a decade after the publication of his paper on statistical self-similarity in the coastline of Britain. One often cited description that he published to describe geometric fractals is 'a rough or fragmented geometric shape that can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole'; this is generally helpful but limited. A fractal is by definition a set for which the *Hausdorff-Besicovitch* dimension strictly exceeds its *topological dimension.* However, since Hausdorff-Besicovitch dimension is often difficult to calculate, the *fractal dimension* is used instead.

An *iterated function system* is a method for constructing fractals and makes the basis of most fractal-based image compression methods. A *fractal interpolation function* can be considered as a continuous function whose graph is the *attractor* of an appropriately chosen iterated function system. If this graph, usually of nonintegral dimension, belongs to the three dimensional space and has Hausdorff–Besicovitch dimension between 2 and 3, then the resulting attractor is called *fractal interpolation surface*.

*Fractal compression* refers to a number of lossy compression methods, based on *fractals*. When applied to digital images, it is called *fractal image compression*. It differs from pixel-based compression schemes such as Joint Photographic Experts Group (JPEG), Graphics Interchange Format (GIF) or Moving Picture Experts Group (MPEG) since no pixels are saved; for comparison's sake see [1]. It is best suited for textures and natural images, relying on the fact that parts of an image often resemble other parts of the same image. Special algorithms convert these parts into mathematical data called *fractal codes* which are used to recreate the encoded and compressed image. These codes can be decoded to fill any screen size without the loss of sharpness that occurs in conventional compression schemes.

The purpose of this article is to overview some basic approaches to fractal-based image encoding and compression in very simple terms, with as little mathematics as possible. Before delving into the main study, we first present the development of an iterated function system. Next, we examine the theoretical formulation of affine fractal interpolation functions as attractors of an appropriately defined iterated function system in two and three dimensions. Moreover, we examine an extension of the affine fractal interpolation functions, namely the piecewise affine fractal interpolation functions. Furthermore, some fundamental fractal compression schemes which use iterated function systems will be presented. Main differences with other fractal-based image compression schemes will be briefly explained. Finally, some conclusions will be drawn. A fairly full picture of the relevant literature is presented in [2] and [3] until the day of their publication. The concepts of fractals, iterated function systems and local iterated function systems are discussed and different implementations of compression of both still images and image sequences are reviewed in [4], [5], [6], [7], [8] and [9]. The bibliography presented here contains the recent bare essentials which are not included in the aforementioned articles.

## 2. Fractal Image Generation

In mathematics, an *iterated function* is a function which is composed with itself, possibly *ad infinitum*, in a process called iteration. *Iteration* means the act of repeating a process with the aim of approaching a desired goal, target or result. The formal definition of an iterated function on a set $X$ follows. Let $X$ be a set and $f$ a function from $X$ to itself i.e. a *transformation*. Define $f^k$ as the $k$-th iterate of $f$, where $k$ is a non-negative integer, by $f^0 = \mathrm{id}_X$ and $f^{k+1} = f \circ f^k$, where $\mathrm{id}_X$ is the identity function on $X$ and $f \circ g$ denotes function composition.

A *contraction mapping,* or *contraction,* on a metric space $(X, \rho)$ is a transformation $f$, with the property that there is a nonnegative real number $s < 1$ such that for all $x$ and $y$ in $X$, $\rho(f(x), f(y)) \leq s \cdot \rho(x, y)$, where $\rho$ is a *distance function* between elements of $X$. The smallest such value of $s$ is called the *Lipschitz constant* or *contractivity factor* of $f$. If the above condition is satisfied for $s \leq 1$, then the mapping is said to be *non-expansive*. A contraction mapping has at most one *fixed point*, i.e. a point $x^*$ in $X$ such that $f(x^*) = x^*$. Moreover, the *Banach fixed point theorem*, also known as the *contraction mapping theorem* or *contraction mapping principle*, states that every contraction mapping on a nonempty, complete metric space has a unique fixed point, and that for any $x$ in $X$ the iterated function sequence $x, f(x), f(f(x)), f(f(f(x))), \ldots$ converges to the fixed point. Furthermore, this fixed point can be found as follows: Start with an arbitrary element $x_0$ in $X$ and define an iterative sequence by $x_n = f(x_{n-1})$ for $n = 1, 2, 3, \ldots$ . This sequence converges and its limit is $x^*$.

An *iterated function system*, or *IFS* for short, is defined as a collection of a complete metric space $(X, \rho)$, e.g. $(\overline{\mathbb{R}}^n, \|\cdot\|)$ or a subset, together with a finite set of continuous transformations $\{w_i : X \to X, i = 1, 2, \ldots, M\}$. It is often convenient to write an IFS formally as $\{X; w_1, w_2, \ldots, w_M\}$ or, somewhat more briefly, as $\{X; w_{1-M}\}$. If $w_i$ are *contractions* with respective contractivity factors $s_i$, $i = 1, 2, \ldots, M$, the IFS is termed *hyperbolic*.

John E. Hutchinson showed in [10] that, for the metric space $\overline{\mathbb{R}}^n$, such a (hyperbolic) system of functions has a unique compact (closed and bounded) *fixed set S*. One way for constructing a fixed set is to start with an initial point or set $S_0$ and iterate the actions of the $w_i$, taking $S_{n+1}$ to be the union of the image of $S_n$ under the $w_i$; then taking $S$ to be the closure of the union of the $S_n$. Symbolically, the unique fixed (nonempty compact) set $S \subset \mathbb{R}$ has the property

$$S = \bigcup_{i=1}^{M} w_i(S).$$

The set $S$ is thus the fixed set of the *Hutchinson operator*

$$H(A) = \bigcup_{i=1}^{M} w_i(A),$$

where $A$ is any subset of $\mathbb{R}^n$. The operator $H$ itself is a contraction with contractivity factor $s = \max\{s_1, s_2, ..., s_M\}$ ([11], Theorem 7.1, p. 81 or [12]). The existence and uniqueness of $S$, which is called the *attractor* of the IFS, is a consequence of the contraction mapping principle as is the fact that $\lim_{k\to\infty} H^k(A) = S \equiv \mathcal{A}_\infty$ for all $A$ in $\mathcal{H}(\mathbb{R}^n)$, where $\mathcal{H}(X)$ is the metric space of all nonempty, compact subsets of $X$ with respect to some metric, e.g. the *Hausdorff metric*. The operator $H$ is also called the *collage map* to alert us to the fact that $H(A)$ is formed as a union or 'collage' of sets.

Fractals derived by IFSs can be of any number of dimensions, but are commonly computed and drawn in 2D. A fractal is made up of the union of several copies of itself, each copy being transformed by a function (hence 'function system'). The canonical example is the *Sierpinski gasket*; see Fig. 1. The functions are normally contractions which means they bring points closer together and make shapes smaller. Hence, such a shape is made up of several possibly-overlapping smaller copies of itself, each of which is also made up of copies of itself, ad infinitum. This is the source of its self-similar nature.

Note that this infinite process is not dependent upon the starting shape being a triangle – it is just clearer that way. The first few steps starting, for example, from a square also tend towards a Sierpinski gasket; see Fig. 2.

Sometimes each function $w_i$ is required to be a linear, or more generally an affine transformation, and hence represented by a matrix. A transformation $w$ is affine, if it may be represented by a matrix $\mathbf{A}$ and translation $\mathbf{t}$ as $w(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{t}$, or, if $X = \mathbb{R}^2$,

$$w\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & s \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d \\ e \end{pmatrix}. \qquad (1)$$

The *code* of $w$ is the 6-tuple ($a, b, c, s, d, e$) and the *code of an* IFS is a table whose rows are the codes of $w_1$, $w_2$, ..., $w_M$. For the three-dimensional case (grey-scale images) this becomes

$$w\begin{pmatrix} x \\ y \\ I(x,y) \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}\begin{pmatrix} x \\ y \\ I(x,y) \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}, \qquad (2)$$

where $I(x, y)$ denotes the grey-scale value at location $(x, y)$. So, the similarity that exists in different parts of an image could be described in a finite set of instructions and data. Using some image-areas of the image, by rotating, resizing and stretching them, it is possible to generate other parts of the image. However, IFSs may also be built from non-linear functions, including *projective* and *Möbius* transformations.

The most common algorithm to compute fractals derived by IFSs is called the chaos game or *random iteration algorithm*. It consists of picking a random point in the plane, then iteratively applying one of the functions chosen at random from the function system and drawing the point. An alternative algorithm, the *deterministic iteration algorithm*, or *DIA* for short, is to generate each possible sequence of functions up to a given maximum length and then to plot the results of applying each of these sequences of functions to an initial point or shape.

## 3. Fractal Interpolation

Based on a theorem of J. E. Hutchinson ([10], p. 731) and using IFS theory [12], M. F. Barnsley introduced a class of functions (see [11]) which he called *fractal interpolation functions*. He worked basically with affine fractal interpolation functions, in the sense that they are obtained using affine transformations. Their main difference from elementary functions is their fractal character. Since their graphs usually have non-integral dimension, they can

*Fig. 1 The evolution of the Sierpinski gasket*

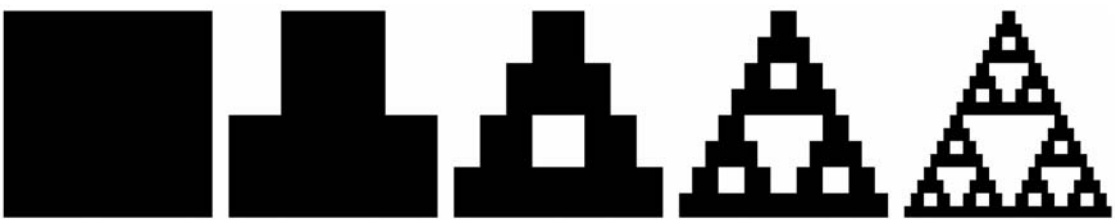

*Fig. 2 Iterating from a square*

be used to approximate image components such as the profiles of mountain ranges, the tops of clouds and horizons over forests, to name but a few. Applications of this theory include modelling of discrete sequences as in [13], modelling of speech signals as in [14] and compression of static images as in Ref. [AlCl94] of [2].

### 3.1 Interpolation Functions in $\mathbb{R}$

Let the continuous function $f$ be defined on a real closed interval $I = [x_0, x_M]$ and with range the metric space $(\mathbb{R}, |\cdot|)$, where $x_0 < x_1 < \cdots < x_M$. It is not assumed that these points are equidistant. The function $f$ is called an interpolation function corresponding to the *generalized set of data* $\{(x_m, y_m) \in K = I \times \mathbb{R} : m = 0, 1, ..., M\}$, if $f(x_m) = y_m$ for all $m = 0, 1, ..., M$ and $K = I \times \mathbb{R}$. The points $(x_m, y_m) \in \mathbb{R}$ are called the *interpolation points*. We say that the function $f$ *interpolates* the data and that (the graph of) $f$ passes through the interpolation points.

Let us represent our, real valued, set of *data points* as $\{(u_n, v_n): n = 0, 1, ..., N; u_n < u_{n+1}\}$ and the interpolation points as $\{(x_m, y_m) : m = 0, 1, ..., M; M \leq N\}$, where $u_n$ is the sampled index and $v_n$ the value of the given point in $u_n$. Let $\{\mathbb{R}^2; w_{1-M}\}$ be an IFS with affine transformations of the special form (see (1))

$$w_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & 0 \\ c_i & s_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_i \\ e_i \end{pmatrix}$$

constrained to satisfy

$$w_i \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} x_{i-1} \\ y_{i-1} \end{pmatrix} \text{ and } w_i \begin{pmatrix} x_M \\ y_M \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{3}$$

for every $i = 1, 2, ..., M$. Solving the above equations results in

$$a_i = \frac{x_i - x_{i-1}}{x_M - x_0}, \; d_i = \frac{x_M x_{i-1} - x_0 x_i}{x_M - x_0},$$

$$c_i = \frac{y_i - y_{i-1}}{x_M - x_0} - s_i \frac{y_M - y_0}{x_M - x_0},$$

$$e_i = \frac{x_M y_{i-1} - x_0 y_i}{x_M - x_0} - s_i \frac{x_M y_0 - x_0 y_M}{x_M - x_0}$$

i.e. the coefficients $a_i$, $c_i$, $d_i$, $e_i$ are completely determined by the interpolation points, while the $s_i$ are free parameters satisfying $|s_i| < 1$ in order to guarantee that the IFS is hyperbolic with respect to an appropriate metric for every $i = 1, 2, ..., M$. The transformations $w_i$ are *shear transformations*: line segments parallel to the y-axis are mapped to line segments parallel to the $y$-axis contracted by the factor $|s_i|$. For this reason, the $s_i$ are called *vertical scaling* (or contractivity) *factors*.

The IFS $\{\mathbb{R}^2; w_{1-M}\}$ has a unique attractor that is the graph of some continuous function which interpolates the data points. This function is called a *fractal interpolation function*, or *FIF* for short, because its graph usually has non-integral dimension. A *section* is defined as the function values between interpolation points. It is a *self-affine* function since each affine transformation $w_i$ maps the entire (graph of the) function to its section. The above function is known as *affine FIF*, or *AFIF* for short. For example, let {(0, 0), (0.4, 0.5), (0.7, 0.2), (1, 0)} be a given set of data points. Fig. 3 shows the graph of an AFIF with $s_1 = 0.5$, $s_2 = -0.2$ and $s_3 = 0.4$. The closeness of fit of a FIF is mainly influenced by the determination of its vertical scaling factors. No direct way to find the optimum values of these factors exists and various approaches have been proposed in the literature.

### 3.2 Piecewise Affine Fractal Interpolation

The *piecewise self-affine* fractal model is a generalization of the affine fractal interpolation model and has its mathematical roots

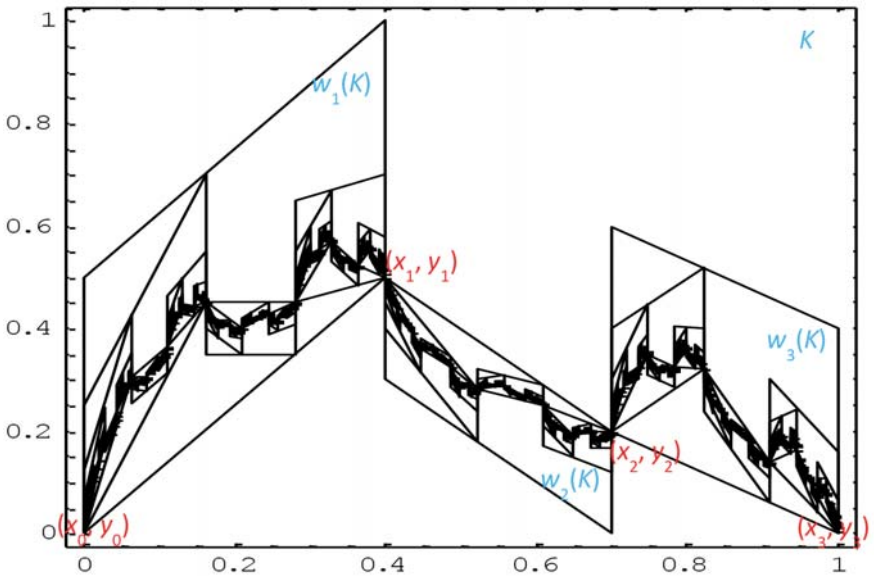


*Fig. 3 The construction of an affine FIF starting from the unit square and using the DIA*

embedded in *recurrent IFS*, or *RIFS* for short, theory. A pair of data points $\{(\tilde{x}_{i,j}, \tilde{y}_{i,j})\colon i = 1, 2, ..., M; j = 1, 2\}$, which are called *addresses* or *address points*, is now associated with each *interpolation interval*. Each pair of addresses defines the *domain* or *address interval*. The constraints (3) become

$$w_i\begin{pmatrix}\tilde{x}_{i,1}\\ \tilde{y}_{i,1}\end{pmatrix} = \begin{pmatrix}x_{i-1}\\ y_{i-1}\end{pmatrix} \text{ and } w_i\begin{pmatrix}\tilde{x}_{i,2}\\ \tilde{y}_{i,2}\end{pmatrix} = \begin{pmatrix}x_i\\ y_i\end{pmatrix}$$

subjected to $\tilde{x}_{i,1} - \tilde{x}_{i,2} > x_i - x_{i-1}$ for every $i = 1, 2, ..., M$. Solving the above equations results in

$$a_i = \frac{x_i - x_{i-1}}{\tilde{x}_{i,2} - \tilde{x}_{i,1}},\ d_i = \frac{\tilde{x}_{i,2}x_{i-1} - \tilde{x}_{i,1}x_i}{\tilde{x}_{i,2} - \tilde{x}_{i,1}},$$

$$c_i = \frac{y_i - y_{i-1}}{\tilde{x}_{i,2} - \tilde{x}_{i,1}} - s_i\frac{\tilde{y}_{i,2} - \tilde{y}_{i,1}}{\tilde{x}_{i,2} - \tilde{x}_{i,1}},$$

$$e_i = \frac{\tilde{x}_{i,2}y_{i-1} - \tilde{x}_{i,1}y_i}{\tilde{x}_{i,2} - \tilde{x}_{i,1}} - s_i\frac{\tilde{x}_{i,2}\tilde{y}_{i,1} - \tilde{x}_{i,1}\tilde{y}_{i,2}}{\tilde{x}_{i,2} - \tilde{x}_{i,1}}$$

for every $i = 1, 2, ..., M$. The function constructed as the attractor of the above-mentioned IFS is called *recurrent fractal interpolation function*, or *RFIF* for short, corresponding to the interpolation points. A RFIF is a piecewise self-affine function since each affine transformation $w_i$ maps the part of the (graph of the) function defined by the corresponding address interval to each section.

### 3.3 Interpolation Functions in $\mathbb{R}^2$

Suppose we refer to an image as a function $z = z(x, y)$ that gives the grey level at each point $(x, y)$. Let the discrete data $\{(x_i, y_j, z_{i,j} = z(x_i, y_j)) \in \mathbb{R}^3 : i = 0, 1, ..., N; j = 0, 1, ..., M\}$ be known. Each affine mapping that comprises the hyperbolic IFS $\{\mathbb{R}^3; w_{1-N,\,1-M}\}$ is given by the following special form of (2)

$$w_{n,m}\begin{pmatrix}x\\ y\\ z\end{pmatrix} = \begin{pmatrix}a_{n.m} & b_{n.m} & 0\\ c_{n.m} & d_{n.m} & 0\\ e_{n.m} & g_{n.m} & s_{n.m}\end{pmatrix}\begin{pmatrix}x\\ y\\ z\end{pmatrix} + \begin{pmatrix}h_{n.m}\\ k_{n.m}\\ l_{n.m}\end{pmatrix},$$

with $|s_{n,m}| < 1$ for every $n = 1, 2, ..., N$ and $m = 1, 2, ..., M$. The condition

$$\left\|\begin{matrix}a_{n,m} & b_{n,m}\\ c_{n,m} & d_{n,m}\end{matrix}\right\| < 1$$

ensures that

$$u_{n,m}\begin{pmatrix}x\\ y\end{pmatrix} = \begin{pmatrix}a_{n,m} & b_{n,m}\\ c_{n,m} & d_{n,m}\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} + \begin{pmatrix}h_{n,m}\\ k_{n,m}\end{pmatrix}$$

is a similitude and the transformed surface does not vanish or flip over.

FIFs are suitable for data sets with points linearly ordered with respect to their abscissa. This is often sufficient, e.g. when interpolating time series data. In practice, however, there are many cases where the data are suitable for fractal interpolation but define a curve rather than a function, e.g. when modelling coastlines or plants. There exist methods for constructing *fractal interpolation curves* based on the theory of FIFs. These methods use various approaches, such as generalizations to higher dimensions, use of index coordinates or application of reversible transformations.

## 4. Fractal Image Encoding and Compression

Fractal encoding relies on the fact that all natural, and most artificial, objects contain redundant information in the form of similar, repeating patterns or fractals. The encoding process is computationally intensive. Depending upon the resolution and contents of the input bitmap data, and output quality, compression time, and file size parameters selected, compressing a single image could take anywhere from a few seconds to a few hours on even a fast computer; see also [15]. Decoding a fractal image is a much simpler process. All the decoding process needs to do is to interpret the fractal codes and translate them into a bitmap image.

The fundamental principle of fractal-based image compression consists of the representation of an image by an IFS of which the fixed point is 'close' to that image. In other words, the encoding process is first to find an IFS and then a suitable operator $H$ whose fixed point is 'close' to the given image.

### 4.1 Based Directly on the Collage Theorem

The *collage theorem* characterizes an IFS whose attractor is close, relative to some metric, to a given set. The IFS described is composed of contractions whose images, as a collage or union when mapping the given set, are arbitrarily close to the given set.

If $B \in (\mathcal{H}(\mathbb{R}^n), h)$, where $h$ is a metric, obeys $h(B, H(B)) \leq \varepsilon$, then

$$h(B, \mathcal{A}_\infty) \leq \frac{\varepsilon}{1 - s},$$

where $s = \max[s_i : i = 1, 2, ..., N]$. Therefore, the closer the union is to the given set, the closer the attractor of the IFS will be to the given set. The theorem, however, is not constructive, it does not indicate how to find a set of proper mappings, but rather, it provides a way to test an IFS without need for computation of the attractor.

To find an IFS for an image, based on the collage theorem and the property of IFS attractors, we split the whole image into non-overlapping segments whose union covers the entire image. If each segment is a transformed copy of the entire image or is very close to it, the combination of these transformations is the IFS of the original image. In other words, to encode an image into IFS is to find a set of contractive affine transformations, $w_1, w_2, ..., w_N$, so that the original image $B$ is the union of the $N$ subimages: $B = w_1(B) \cup w_2(B) \cup \cdots \cup w_N(B)$.

### 4.2 Based on the Fractal Transform

The fractal transform [16] was considered to solve effectively the problem of finding a fractal which approximates a digital 'real world image.' This first practical fractal compression system for digital images resembles a vector quantization system using the image itself as the codebook.

**Fractal Transform compression**

Start with a digital image $A_1$. Downsample (subsample) it by a factor of 2 to produce image $A_2$. Now, for each block $B_1$ of 4×4 pixels in $A_1$, find the corresponding block $B_2$ in $A_2$ most similar to $B_1$ and then find the grey-scale or RGB offset and gain from $A_2$ to $B_2$. For each destination block, output the positions of the source blocks and the color offsets and gains.

An image compressed in this way contains a minimal area plus a transformation matrix that contains all required information to reconstruct something similar to the original image by rotating, resizing and stretching this area. Therefore the task of fractal encoding algorithms is to find redundant areas of an image and to reduce these areas to the necessary information about attractor and transformation matrix. To perform this task, the image is divided into *domains* in coarse or fine resolution depending on the redundancy of the image. Then the routine searches the image for *ranges* that have a similarity to the domain that can be described by an affine transformation matrix. The names of these blocks are reversed in Ref. [7, p. 181] of [3].

**Fractal Transform decompression**

Starting with an empty destination image $A_1$, repeat the following algorithm several times: Downsample $A_1$ down by a factor of 2 to produce image $A_2$. Then copy blocks from $A_2$ to $A_1$ as directed by the compressed data, multiplying by the respective gains and adding the respective color offsets.

This algorithm is guaranteed to converge to an image, and it should appear similar to the original image. In fact, a slight modification of the decompressor to run at block sizes larger than 4×4 pixels produces a method of stretching images without causing the blockiness or blurriness of traditional *linear resampling algorithms.*

### 4.3 Based on Local (Partitioned) IFS

IFS approximates each part of the image by a transformed version of the *whole* image, but our intuition suggests that real-world images, generally, do not contain parts that are affine transforms of the whole image (Fig. 4). Nevertheless, different parts of the image may become similar under certain affine transformation. Therefore, an extension of the IFS, the *Local* or *Partitioned* IFS, has been developed to approximate each part of the image by a transformed version of *another part* of the image. The first automated compression scheme for real world images using PIFS was developed by Arnaud E. Jacquin (Ref. [26] of [3]) in 1992.

Suppose we are given an image $f$ we wish to encode. We divide the image into *range blocks* $R_1$, $R_2$, ..., $R_i$, ..., $R_N$, such that $f = R_1 \cup R_2 \cup \cdots \cup R_N$ and $R_i \cap R_j = \varnothing$ when $i \neq j$. Therefore, the range blocks cover the whole image and do not overlap. The image is also divided into overlapping *domain blocks* $D_1$, $D_2$, ..., $D_j$, ..., $D_M$. For each range block $R_i$, we find a contractive transformation wi and a domain block $D_j$, so that $R_i \approx w_i(D_j)$. The combination of $w_1$, $w_2$, ..., $w_i$, ..., $w_N$ is called PIFS $H$. If $H$ is simpler than the original image, we can encode $f$ into $H$ and achieve certain compression. When decoding, according to the contractive mapping fixed point theorem, as long as $H$ is contractive, repeated application of $H$ to an arbitrary image will result in a fixed image. When $H(f)$ is close to $f$, the fixed image will be close to the original image $f$.

The three main issues involved in the design and implementation of a fractal block-coding system based on the above idea are (i) how the image is partitioned, (ii) the choice of a distortion measure between two images and (iii) types of contractive affine transformations to be used. We do not intend to discuss these

*Fig. 4 The original 256 × 256 × 8 bpp image of Lenna (left) and self-similar portions of Lenna's image (right)*

issues; see [17] for highlighting the solution to the key and open issues in the fractal coding. It is noteworthy to mention, though, how important is to find a similar block so that the IFS accurately represents the input image, so a sufficient number of candidate blocks for $D_i$ need to be considered. On the other hand, a large search considering many blocks is computationally costly. This bottleneck of searching for similar blocks is why fractal encoding is slower than for example DCT, the underlying compression technique in JPEG, and wavelet based image representations.

### 4.4 Based on RIFS or on FIF

A RIFS is an improvement of LIFS using elements of the theory of *Marcovian stochastic processes* which can produce more natural looking images; see [18]. An image compression scheme using fractal interpolation surfaces which are attractors of some RIFSs is introduced in [19]. Alternative image compression methods which use the method presented in Subsection 3.3 can be found in [20]. Many enhancements and variations exist, as for example [21], but they fall beyond the scope of this discussion.

The resolution independence of a fractal-encoded image can be used to increase the display resolution of an image. During this process, also known as fractal interpolation, an image is encoded into fractal codes via fractal compression and subsequently decompressed at a higher resolution. The result is an up-sampled image in which an appropriately chosen iterated function system has been used as the interpolant. Because fractal interpolation operates on geometric information in the image, rather than pixel information, it maintains geometric detail very well compared to other interpolation methods.

## 5. Conclusions

Our clarification of terminology used in the existing literature led to the conclusion that fractal image compression is also called as fractal image encoding because a compressed image is represented by contractive transformations and mathematical functions required for reconstructing the original image.

Lossless compression methods are sometimes preferred for artificial images such as technical drawings, icons or comics. This is because lossy compression, especially when used at low bit rates, introduce compression artifacts. Lossless compression may also be preferred for high value content, such as medical imagery or image scans made for archival purposes. Lossy methods are especially suitable for natural images such as photos in applications where minor loss of fidelity is acceptable to achieve a substantial reduction in bit rate. For further details see [22] and [23].

Fractal image compression enables an incredible amount of data to be stored in highly compressed data files. An inherent feature of fractal compression is that images become resolution independent after being converted to fractal code. This is because the iterated function systems in the compressed file scale indefinitely. This indefinite property, known as *fractal scaling* or *fractal zooming*, leaves no trace of the original pixel structure.

**Acknowledgement**

The author would like to thank Prof. Mariana Marčoková and Prof. Ing. Branislav Dobrucky for the invitation to write this article based on the lectures he gave two years ago at the University of Zilina. He also wishes to thank both reviewers for their comments.

## References

[1] RUSS , J. C., *The Image Processing Handbook*, 6th ed., CRC Press LLC, 2011.
[2] SAUPE, D., HAMZAOUI, R.: A Review of the Fractal Image Compression Literature, *ACM SIGGRAPH Computer Graphics,* vol. 28, No. 4, 1994, pp. 268–276.
[3] WOHLBERG, B., DE JAGER, G.: A Review of the Fractal Image Coding Literature, *IEEE Trans. Image Process.,* vol. 8, No. 12, 1999, pp. 1716–1729.
[4] CHAURASIA, V., SOMKUWAR, A.: Review of a Novel Technique: Fractal Image Compression, *Int. J. Emerging Technol.*, vol. 1, No. 1, 2010, pp. 53–56.
[5] CHAUDHARI, R. E., DHOK, S. B.: Review of Fractal Transform Based Image and Video Compression, *Int. J. Comput. Appl.,* vol. 57, No. 19, 2012, pp. 23–31.
[6] DHAWAN, S.:A Review of Image Compression and Comparison of its Algorithms, *Int. J. of Electronics & Commun. Technol.,* vol. 2, No. 1, 2011, pp. 22–26.
[7] GHARAVI-ALKHANSARI, M., HUANG, T. S.:, Fractal-based Image and Video Coding, in L. Torres and M. Kunt (eds.), *Video coding:* The second generation approach, pp. 265-303. Kluwer Academic Publishers : Boston, 1996.
[8] LU, G.: Fractal-based Image and Video Compression, in K. R. Rao and P. C. Yip (eds.), *Chapter 7: The transform and data compression handbook,* CRC Press LLC, Boca Raton, 2001.
[9] ZHAO, E., LIU, D.: Fractal Image Compression Methods: A Review, *Proc. IEEE Int. Conf. Inf. Technol. and Appl.*, vol. 1, 2005, pp. 756–759.
[10] HUTCHINSON, J. E.: Fractals and Self Similarity, *Indiana Univ. Math. J.*, vol. 30, No. 5, 1981, pp. 713–747.
[11] BARNSLEY, M. F.: *Fractals Everywhere,* 2nd ed., Academic Press Professional, 1993.

[12] BARNSLEY, M. F., DEMKO, S.: Iterated function systems and the global construction of fractals, *Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, vol. 399, No. 1817, 1985, pp. 243–275.
[13] MAZEL, D. S., HAYES, M. H.: Using Iterated Function Systems to Model Discrete Sequences, *IEEE Trans. Signal Process.,* vol. 40, No. 7, 1992, pp. 1724–1734.
[14] VEHEL, J. L., DAOUDI, K., LUTTON, E.: Fractal Modeling of Speech Signals, *Fractals,* vol. 2, No. 3, 1994, pp. 379–382.
[15] FURAO, S., HASEGAWA, O.: A Fast no Search Fractal Image Coding Method, *Signal Process.: Image Commun.*, vol. 19, No. 5, 2004, pp. 393–404.
[16] BARNSLEY. M. F., Anson, L. F.: *The Fractal Transform,* Jones and Bartlett Publishers, Inc, 1993.
[17] Koli, N. A., Ali, M. S.: A Survey on Fractal Image Compression Key Issues. *Inf. Technol. J.*, vol. 7, No. 8, 2008, pp. 1085–1095.
[18] HURT, J. C. : Fractal Image Compression and Recurrent Iterated Function Systems, *IEEE Comput. Graphics & Appl.*, vol. 16, No. 4, 1996, pp. 25–33.
[19] BOUBOULIS, P., DALLA, L., DRAKOPOULOS, V.: Image Compression Using Recurrent Bivariate Fractal Interpolation Surfaces, *Int. J. Bifurc. Chaos,* vol. 16, No. 7, 2006, pp. 2063–2071.
[20] DRAKOPOULOS, V., BOUBOULIS, P., THEODORIDIS, S.: Image Compression Using Affine Fractal Interpolation on Rectangular Lattices, *Fractals,* vol. 14, No. 4 2006, pp. 259–269.
[21] CHERNOYAROV, O. V. BREZNAN, M.: Optimal and Quasioptimal Algorithms of Distinction of the Compressed Images in Base of Orthogonal Polynomials, *Communications – Scientific Letters of the University of Zilina,* vol. 14, No. 2, 2012, pp. 22–26.
[22] BARNSLEY, M. F., SAUPE, D., VRSCAY, E. R. (eds.): *Fractals in Multimedia,* Springer-Verlag : New York, 2002.
[23] NIKIEL, S.: *Iterated Function Systems for Real-Time Image Synthesis,* Springer-Verlag : London, 2007.