Applied Mathematics, 2014, 5, 1810-1819
Published Online June 2014 in SciRes. http://www.scirp.org/journal/am
http://dx.doi.org/10.4236/am.2014.512174

Scientific Research

Fractal Image Compression Using Self-Organizing Mapping

Rashad A. Al-Jawfi[1,2], Baligh M. Al-Helali[1], Adil M. Ahmed[3]

[1]Department of Mathematics and Computer Science, Faculty of Science, Ibb University, Ibb, Yemen
[2]Department of Mathematics, Faculty of Sciences and Arts, Najran University, KSA
[3]Department of Mathematics, Faculty of Ibn Alhaitham for Education, Baghdad University, Baghdad, Iraq
Email: algofi@yemen.net.ye, baleegh.helali@gmail.com, adilm.ahmed@yahoo.com

Received 25 January 2014; revised 1 March 2014; accepted 9 March 2014

Abstract

One of the main disadvantages of fractal image data compression is a loss time in the process of image compression (encoding) and conversion into a system of iterated functions (IFS). In this paper, the idea of the inverse problem of fixed point is introduced. This inverse problem is based on collage theorem which is the cornerstone of the mathematical idea of fractal image compression. Then this idea is applied by iterated function system, iterative system functions and grayscale iterated function system down to general transformation. Mathematical formulation form is also provided on the digital image space, which deals with the computer. Next, this process has been revised to reduce the time required for image compression by excluding some parts of the image that have a specific milestone. The neural network algorithms have been applied on the process of compression (encryption). The experimental results are presented and the performance of the proposed algorithm is discussed. Finally, the comparison between filtered ranges method and self-organizing method is introduced.

Keywords

Fractal Image Compression, Organizing Mapping

1. Introduction

The mathematics behind fractals began to take shape in the 17th century when mathematician and philosopher Leibniz considered recursive self-similarity (although he made the mistake of thinking that only the straight line was self-similar in this sense) [1]. Iterated functions in the complex plane were investigated in the late 19th and early 20th centuries by Henri Poincar, Felix Klein, Pierre Fatou and Gaston Julia. However, without the aid of modern computer graphics, they lacked the means to visualize the beauty of many of the objects that they had

How to cite this paper: Al-Jawfi, R.A., Al-Helali, B.M. and Ahmed, A.M. (2014) Fractal Image Compression Using Self-Organizing Mapping. *Applied Mathematics*, **5**, 1810-1819. http://dx.doi.org/10.4236/am.2014.512174

discovered [2]. In the 1960s, Benot Mandelbrot started investigating self-similarity in papers such as How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension, which built on earlier work by Lewis Fry Richardson. Finally, in 1975 Mandelbrot coined the word fractal to denote an object whose Hausdorff-Besicovitch dimension is greater than its topological dimension [2]. Fractal image compression (FIC) was introduced by Barnsley and Sloan [3]. They introduce in another work a better way to compress images [4], and after that, (FIC) has been widely studied by many scientists. FIC is based on the idea that any image contains self-similarities, that is, it consists of small parts similar to itself or to some big part in it [5]. So in FIC iterated function systems are used for modeling. Jacquin [6] presented a more flexible method of FIC than Barnsley's, which is based on recurrent iterated function systems (RIFSs) introduced first by him. RIFSs which have been used in image compression schemes consist of transformations which have a constant vertical contraction factor. Fisher [7] improved the partition of Jacquin. A hexagonal structure called the Spiral Architecture (SA) [8] was proposed by Sheridan in 1996. Bouboulis *et al.* [9] introduced an image compression scheme using fractal interpolation surfaces which are attractors of some RIFSs. Kramm presented a quite fast algorithm [10], manages to merge low-scale redundancy from multiple images. In this work, Neural Networks is used to optimize the process, by using self-organizing neural networks to provide domain classification. The experimental results are presented and the performance of the algorithms is discussed.

Artificial Neural Networks (ANN) has been used for solving many problems, special in cases where the results are very difficult to achieve by traditional analytical methods. There have already been a number of studies published applying ANN to image compression [11]. It is important to emphasize, although there is no sign that neural networks can take over the existing techniques [11], research on neural networks for image compression is still making some advances. Possibly in the future this could have a great impact on the development of new technologies and algorithms in this area.

J. Stark first proposed a research to apply the neural network to iterated function system (IFS) [12]. His method was using Hopfield neural network to solve the linear progressive problem and get the Hutchinson metric quickly. However, his neural network approach cannot obtain the fractal code automatically. A few methods of optimization of exhaustive search [13] [14], which were based on clustering of the set of domain blocks, were suggested. But the majorities of these methods either decrease lightly the computational complexity or result in high losses of the quality of an image. The method of clustering by means of Artificial Kohonen neural self-optimizing network is least afflicted with these disadvantages [15].

2. Neural Networks

A neural net is an artificial representation of the human brain that tries to simulate its learning process. The term "artificial" means that neural nets are implemented in computer programs that are able to handle the large number of necessary calculations during the learning process. To show where neural nets have their origin, let's have a look at the biological model: the human brain.

2.1. The Components of a Neural Net

Generally spoken, there are many different types of neural nets, but they all have nearly the same components. If one wants to simulate the human brain using a neural net, it is obviously that some drastic simplifications have to be made: First of all, it is impossible to "copy" the true parallel processing of all neural cells. Although there are computers that have the ability of parallel processing, the large number of processors that would be necessary to realize it can't be afforded by today's hardware. Another limitation is that a computer's internal structure can't be changed while performing any tasks.

And how to implement electrical stimulations in a computer program? These facts lead to an idealized model for simulation purposes. Like the human brain, a neural net also consists of neurons and connections between them. The neurons are transporting incoming information on their outgoing connections to other neurons. In neural net terms these connections are called weights. The "electrical" information is simulated with specific values stored in those weights. By simply changing these weight values the changing of the connection structure can also be simulated.

As you can see, an artificial neuron looks similar to a biological neural cell. And it works in the same way, input is sent to the neuron on its incoming weights. This input is information called the propagation function that adds up the values of all incoming weights. processed by a threshold value by the neuron's activation function. The resulting value is compared with a certain if the input exceeds the threshold value, the neuron will be

activated, otherwise it will be inhibited. output on its outgoing weights to all connected neurons and so if activated, the neuron sends an on. **Figure 1(a)** shows a neural net structure. In a neural net, the neurons are grouped in layers, called neuron layers. Usually each neuron of input one layer is connected to all neurons of the preceding and the following layer (except the layer and the output layer of the net). The information given to a neural net is propagated layer-by-layer from input layer to output layer hidden layers. Depending on the learning algorithm, it is also through either none, one or more possible that information is propagated backwards through the net.

Figure 1(b) shows a neural net with three neuron layers.

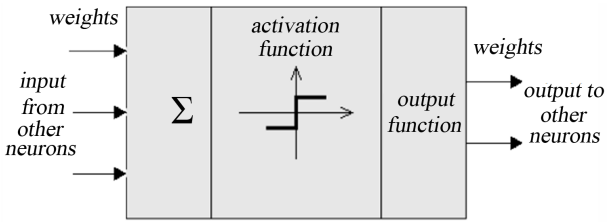
Note that this is not the general structure of a neural net. For example, some neural net types have no hidden layers or the neurons in a layer are arranged as a matrix, weight matrix, the what's common to all neural net types is the presence of at least one connections between two neuron layers.

2.2. Types of Neural Nets

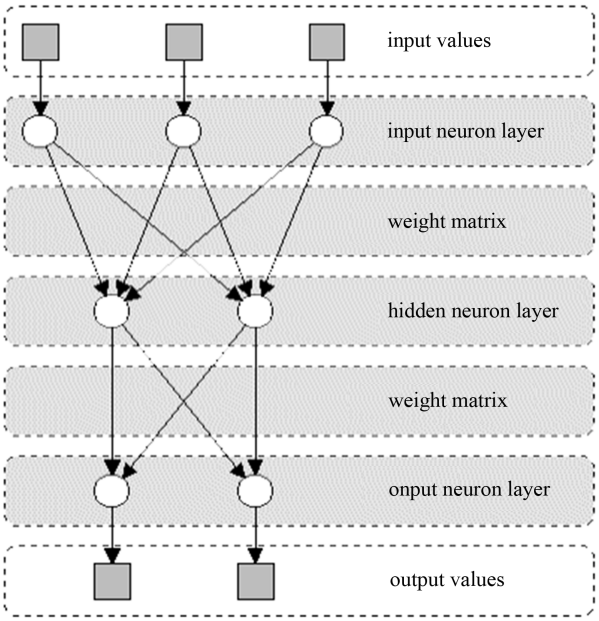
As mentioned before, several types of neural nets exist. They can be distinguished by their type (feedforward or feedback), their structure and the learning algorithm they use. The type of a neural net indicates, if the neurons of one of the net's layers may be connected among each other. Feedforward neural nets allow only neuron connections between two different layers, while nets of the feedback type have also connections between neurons of the same layer.

2.3. Supervised and Unsupervised Learning

Neural nets that learn unsupervised have no such target outputs. It can't be determined what the result of the



(a)



(b)

Figure 1. (a) Structure of a neuron in a neural net. (b) Neural net with three neuron layers.

learning process will look like. During the learning process, the units (weight values) of such a neural net are "arranged" inside a certain range, depending on given input values. The goal is to group similar units close together in certain areas of the value range. This effect can be used efficiently for pattern classification purposes.

3. The Self-Organizing Mapping (SOM)

3.1. Competitive Learning and Clustering

Competitive learning is a learning procedure that divides a set of input patterns in clusters that are inherent to the input data. A competitive learning network is provided only with input vectors x and thus implements an unsupervised learning procedure. We will show its equivalence to a class of 'traditional' clustering algorithms shortly.

3.2. Winner Selection: Euclidean Distance

In the competitive structures, a winning processing element is determined for each input vector based on the similarity between the input vector and the weight vector.

To this end, the winning neuron k is selected with its weight vector w_k closest to the input pattern x, using the Euclidean distance measure:

$$k : \|w_k(t) - x\| \le \|w_l(t) - x\|, \forall l. \tag{1.1}$$

the winning unit can be determined by

$$\|w_k(t) - x\| = \min_l \|w_l(t) - x\| \tag{1.2}$$

where the index k refers to the winning unit.

Once the winner k has been selected, the weights are updated according to:

$$w_k(t+1) = w_k(t) + \gamma\left(x(t) - w_k(t)\right). \tag{1.3}$$

Note that only the weights of winner k are updated. The weight update given in Equation (1.20) effectively implement a shift to the weight vector w_l towards the input vector x.

3.3. Cost Function

Earlier it was claimed, that a competitive network performs a clustering process on the input data, *i.e.*, input patterns are divided in disjoint clusters such that similarities between input patterns in the same cluster are much bigger than similarities between inputs in different clusters. Similarity is measured by a distance function on the input vectors, as discussed before. A common criterion to measure the quality of a given clustering is the square error criterion, given by

$$E = \sum_p \|w_k - x^p\|^2, \tag{1.4}$$

where k is the winning neuron when input x^p is presented. The weights w are interpreted as cluster centers. It is not difficult to show that competitive learning indeed seeks to find a minimum for this square error by following the negative gradient of the error-function [16]:

Theorem 3.1 *The error function for pattern* x^p *is*

$$E^p = \sum_i \left(w_{ki} - x_i^p\right)^2, \tag{1.5}$$

where k *is the winning unit, is minimized by the weight update rule*

$$w_k(t+1) = w_k(t) + \gamma\left(x(t) - w_k(t)\right). \tag{1.6}$$

Proof 1 *We calculate the effect of a weight change on the error function. So we have that*

$$\Delta_p w_{il} = -\gamma \frac{\partial E^p}{\partial w_{il}}. \tag{1.7}$$

where γ *is a constant of proportionality. Now, we have to determine the partial derivative of* E^p:

$$\frac{\partial E^p}{\partial w_{il}} = \begin{cases} w_{il} - x_i^p, & \text{if } l \text{ wins} \\ 0 & \text{otherwise,} \end{cases} \tag{1.8}$$

such that

$$\Delta_p w_{il} = -\gamma\left(w_{il} - x_i^p\right) = \gamma\left(x_i^p - w_{il}\right) \tag{1.9}$$

which is Equation (1.6) *written down for one element of* w_l. *Therefore, Equation* (1.4) *is minimised by repeated weight updates using Equation* (1.6).

3.4. Winner Selection: Dot Product

For the time being, we assume that both input vectors x and weight vectors w_l are normalised to unit length. Each output unit l calculates its activation value y_l according to the dot product of input and weight vector:

$$y_l = \sum_i w_{il} x_i = W_l^{\mathrm{T}} X. \tag{1.10}$$

In a next pass, output neuron k is selected with maximum activation

$$\forall l \neq k : y_l \leq y_k. \tag{1.11}$$

Activations are reset such that $y_k = 1$ and $y_{l \neq k} = 0$.

This is the competitive aspect of the network, and we refer to the output layer as the winner-take-all layer. The winner-take-all layer is usually implemented in software by simply selecting the output neuron with highest activation value.

We now prove that Equation (1.1) reduces to (1.10) and (1.11) if all vectors are normalised.

Proposition 3.2 *Let* x, *and* w_l *be a normalised vectors, and let* w_k *be selected such that*

$$\left\|w_k(t) - x\right\| \leq \left\|w_l(t) - x\right\|, \forall l \quad \text{then} \quad y_l \leq y_k : \forall l \neq k,$$

where $y_l = \sum_i w_{il} x_i$.

Proof 2 *Let* x *be a normalised input vector and* w_k *be the winning unit is determined by* 1.1 *the minimum of the quantity* $\left\|w_l(t) - x\right\|$, *i.e.*

$$\left\|w_k(t) - x\right\| \leq \left\|w_l(t) - x\right\|, \forall l$$

where $\|y - x\| = \sum_i (y_i - x_i)^2$,
then, we have that,

$$w_k(t) - x \leq w_l(t) - x \Rightarrow \sum_i (w_{ki} - x_i)^2 \leq \sum_i (w_{li} - x_i)^2, \forall l \tag{1.12}$$

$$\Rightarrow \sum_i \left(w_{ki}^2 - 2w_{ki}x_i + x_i^2\right) \leq \sum_i \left(w_{li}^2 - 2w_{li}x_i + x_i^2\right), \forall l \tag{1.13}$$

$$\Rightarrow \sum_i w_{ki}^2 - \sum_i 2w_{ki}x_i + \sum_i x_i^2 \leq \sum_i w_{li}^2 - \sum_i 2w_{li}x_i + \sum_i x_i^2, \forall l \tag{1.14}$$

$$\Rightarrow 1 - \sum_i 2w_{ki}x_i + 1 \leq 1 - \sum_i 2w_{li}x_i + 1, \forall l \tag{1.15}$$

$$\text{where } \sum_i w_{ki}^2 = \sum_i x_i^2 = 1 \tag{1.16}$$

$$\Rightarrow -2\sum_i w_{ki}x_i \leq -2\sum_i w_{li}x_i, \forall l \tag{1.17}$$

$$\Rightarrow \sum_i w_{ki}x_i \geq \sum_i w_{li}x_i, \forall l \tag{1.18}$$

$$\Rightarrow y_k \geq y_l, \forall l, \tag{1.19}$$

where $y_l = \sum_i w_{li}x_i$, *and* $y_k = \sum_i w_{ki}x_i$.

The Euclidean distance norm is therefore a more general case of Equations (1.10) and (1.11).

It can be shown that this network converges to a situation where only the neuron with highest initial activation survives, whereas the activations of all other neurons converge to zero. Once the winner k has been selected, the weights are updated according to:

$$w_k(t+1) = \frac{w_k(t) + \gamma\left(x(t) - w_k(t)\right)}{w_k(t) + \gamma\left(x(t) - w_k(t)\right)}, \tag{1.20}$$

where the divisor ensures that all weight vectors w are normalised. Note that only the weights of winner k are updated. The weight update given in Equation (1.20) effectively rotates the weight vector w_l towards the input vector x. Each time an input x is presented, the weight vector closest to this input is selected and is subsequently rotated towards the input. Consequently, weight vectors are rotated towards those areas where many inputs appear: the clusters in the input.

Previously it was assumed that both inputs x and weight vectors w were normalised. Using the activation function given in Equation (1.10) gives a "biological plausible" solution. **Figure 2(b)** shown how the algorithm would fail if unnormalised vectors were to be used.

An almost identical process of moving cluster centres is used in a large family of conventional clustering algorithms known as square error clustering methods, e.g., *k*-means, forgy, isodata, cluster. From now on, we will simply assume a winner k is selected without being concerned which algorithm is used.

4. The Inverse Problem of Fractals

The term Fractals was coined by Mandelbrot in 1975 to such sets, from the Latin word fractus, meaning broken. He provided a precise technical definition [17]: "fractal is a set with Hausdorff dimension strictly greater than its topological dimension". Instead of giving a precise definition of fractals which almost exclude some interesting cases, it is better to regard a fractal as a set that has the properties [18]:

1) Has a "fine" structure.
2) Has some type of self-similarity.
3) Difficult to be described globally or locally by the classic Euclidean geometry.
4) Usually has a non-integer dimension.
5) Defined by a simple model that can be rendered recursively or iteratively.
6) Is usually a strange attractor for a dynamical system.

4.1. Iterated Function System

Barnsleyin 1988 introduced the iterated function system (IFS) [17] as an applications of the theory of discrete dynamical systems and useful tools to build fractals and other self-similar sets. The mathematical theory of IFS is one of the basis for modeling techniques of fractals and is a powerful tool for producing mathematical fractals

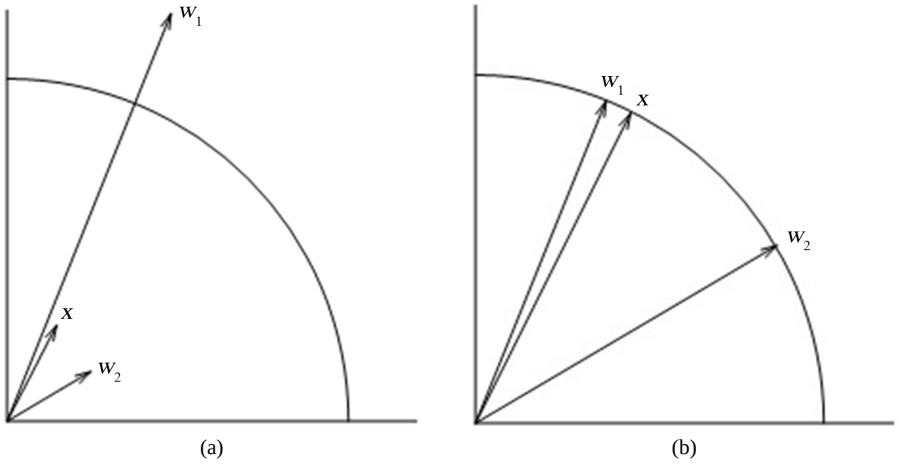


Figure 2. (a) The selection failed in (b) if unnormalised.

such as Cantor set, Sirpinski gasket, etc, as well as real word fractals representing such as clouds, trees, faces, etc. IFS is defined through a finite set of affine counteractive mapping mostly of the form:

$f_i : R^n \to R, i = 1, 2, \cdots, n, n \in N$ In particular case, two-dimensional affine maps have the following form:

$$f\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}.$$

This map could be charaterized by the six constants *a*, *b*, *c*, *d*, *e*, *f*, which establish the code of *f*.

4.2. Fractal Inverse Problem

The fractal inverse problem is an important research area with a great number of potential application fields. It consists in finding a fractal model or code that generates a given object. This concept has been introduced by Barnsley with the well known collage theorem [5]. When the considered object is an image, we often speak about fractal image compression. A method has been proposed by Jacquin [6] to solve this kind of inverse problem.

4.3. Collage Theorem on $(\mathcal{H}(X), h)$

Since the number of points in fractal sets is infinite and complicatedly organized, it is difficult to specify exactly the generator IFS. From the practical point of view, it will be acceptable for the required IFS to be chosen such that its attractor is close to a given image for a pre-defined tolerance.

The collage theorem is very useful to simplify the inverse problem for fractal images [5], it has been addressed by many researchers as well [19].

Theorem 4.1 *Let* $\{X; w_1, w_2, \cdots, w_N\}$ *be a hyperbolic IFS with contractivity factor s, and W be the associated hutchinson map then*

$$h(A, A_W) < \frac{h(A, W(A))}{1-s}, \ \forall A \in \mathcal{H}(X) \tag{1.21}$$

where A_W *is the fixed point of* W.

Hence if $h(A, W(A)) < \epsilon$ *then*

$$h(A, A_W) < \frac{\epsilon}{1-s}.$$

Proof 3 *see* [5].

The theorem can be used as following. Given a fractal image A, find a set of contractive mappings that maps A into smaller copies of itself such that the union of the smaller copies is close as ε to the target image. The determined contractions are the IFS codes with corresponding Hutchinson operator W.

The theorem states that, the attractor A_W of the determined IFS W approximates the target image A (*i.e.*, $h(A, A_W) < \frac{\epsilon}{1-s}$. It also implies that, the more accurately the IFS maps the image to itself, the more accurately the IFS approximates the image.

5. Fractal Image Compression by Means of Kohonen Network

The Kohonen layer is a Winner-take-all (WTA) layer. Thus, for a given input vector, only one Kohonen layer output is 1 whereas all others are 0. No training vector is required to achieve this performance. Hence, the name: Self-Organizing Map Layer (SOM-Layer).

Let $r_i \in R^n$ is a vector of the intensity of the range-block R_i, and $d_j \in \mathbb{R}^n$ is a vector of the intensity of the range-block D_j which is transformed to the size of the corresponding range-block:

$$E(r_i, d_j) = \min_{\alpha,\beta} \left\| r_i - (\alpha d_j + \beta C) \right\|, \ (\alpha, \beta) \in \mathbb{R}^2,$$

where $C \in \mathbb{R}^n, C = (1, \cdots, 1)/\sqrt{n}$. Let O be an operator of orthogonal projection which projects $\mathbb{R}^n$ on the orthogonal complement $\Gamma^{\perp}, \Gamma$ is a linear envelope of the vector C, for $Z = (z_1, \cdots, z_n) \in \mathbb{R}^n \setminus \Gamma$ we shall

define the operator:

$$\tau(Z)=\frac{OZ}{OZ}.$$

Theorem 5.1 *Assume that* $n\geq 2$ *and* $X=\mathbb{R}^n\setminus\Gamma$. *Let us define the function* $\Delta X\times X\to\left[0,\sqrt{2}\right]$ *in the following way*:

$$\Delta(d,r)=\min\left(\left\|\tau(r)+\tau(d)\right\|,\left\|\tau(r)-\tau(d)\right\|\right).$$

For $r_i,d_j\in X$ *the minimum distance* $E\left(r_i,d_j\right)$ *will be determined by the formula*:

$$E\left(r_i,d_j\right)=\left(r_i,\tau\left(r_i\right)\right)g\left(\Delta\left(r_i,d_j\right)\right),$$

where

$$g(\Delta)=\Delta\sqrt{1-\frac{\Delta^2}{4}}.$$

Proof 4 *See* [20].

This theorem means that the less of the distance $d\left(\tau\left(r_i\right),\pm\tau\left(d_j\right)\right)$ the less of the Error $E\left(r_i,d_j\right)$. Let $D_\tau=\left\{\pm\tau\left(d_j\right),\forall j\right\}$ and $R_\tau=\left\{\tau\left(r_i\right),\forall i\right\}$, which we call range--vector and domain--vector receptively.

Let us apply the kohonen network algorithm to cluster the domain vectors. In the beginning, domain vectors will train the network (learning), and, later, the range vectors will input the network to choose the optimal domain.

5.1. Global Codebook

The idea of the global codebook is to assign a fixed domain pool for the entire range pool or for a specific class of it (e.g. set of range blocks that have the same size in a quad tree partition) [16]. In the global codebook each rang block in the range pool has its own domain pool, and this domain pool can be selected by many ways. One of the methods of selecting the domain pool for a range block is to construct a set of domain blocks that are spatially close to the range block [11].

5.2. Ranges Filtering Algorithm

1) set i = 0.
2) if $\left(var\left(r_i\right)>\epsilon\right)$ goto 5.
3) set the arguments of this range (position and the mean $\overline{r_i}$ are enough) to the first codebook.
4) omit r_i from R and mark it as one typed.
5) $i=i+1$.
6) if $i<N_r$ goto 2.
7) End.

5.3. (Training) Domains Clustering Algorithm

1) Network initialization. Let the neurons of the net are

$$Dc=\left\{dc_i\right\}_{i=0}^{Nc}$$

where dc_i is the cluster (neuron) number i of Nc neurons and w_i is the weight vector of the (neuron) number i which initialized to random domain.

2) Search the nearest cluster for $\tau\left(d_j\right)\in D_\tau$. chose the winner neuron $dc_k\in Dc$. $\forall\tau\left(d_j\right)\in D_\tau$ find w_k such that:

$$d\left(w_k,\tau\left(d_j\right)\right)\leq d\left(w_i,\tau\left(d_j\right)\right),\ i=1,\cdots,Nc$$

and add index of the vector $\tau\left(d_j\right)$ into the corresponding memory Dc_k.

3) Update the weight vector w_k of the winning neuron dc_k by the following way:

Table 1. Results of the classic algorithm (without neural networks) vs the neurofractal.

M Image	Encoding Time (sec)		PSNR(dB)	
	Filtered FIC	Neuro FIC	FIC	Neuro FIC
Constant	27	7	54.4	54.1
Lenna	187	185	30.1	28.8
Airplane	166	122	30.4	29.6

$$w_k = w_k + \gamma\left(\tau\left(d_j\right) - w_k\right), \gamma \in (0,1)$$

4) End.

5.4. (Encoding) Ranges Matching Algorithm

1) for each cluster dc_k let $min_k = \min_{\forall j \in Dc_k} d\left(w_k, \tau\left(d_j\right)\right)$ and $max_k = \max_{\forall j \in Dc_k} d\left(w_k, \tau\left(d_j\right)\right)$.
2) $\forall r \in R$ set $k = 0$, $dr = d\left(\tau\left(r\right), w_k\right)$.
3) let $dr_k = d\left(\tau\left(r\right), w_k\right)$.
4) $\forall j \in Dc_k$ if $d\left(\tau\left(r\right), \tau\left(d_j\right)\right) \le dr$ then $r_{code} = arg\left(\tau\left(d_j\right)\right)$ and $dr = d\left(\tau\left(r\right), \tau\left(d_j\right)\right)$.
5) $k = k + 1$ if $k > Nc$ then goto 7.
6) if $d_r - max_k \le d_r$ goto 3 else goto 5.
7) set the r_{code} to the codebook.
8) End.

6. Results and Discussion

A gray level images of size 256×256 have been considered for training the network. A range pool is created having ranges of size 4×4 and $8 \times *$ domain blocks.

The computer simulations have been carried out in Visual *C#* environment on Pentium Dual CPU with 1.73 GHz and 2.00 GB RAM and the results have been presented in **Table 1**.

Table 1 compares fractal image compression results where the standard scheme is introduced in [19] and the self-organizing method. The neural method classifies domains using the self-organizing neural network approach, in each case, a total of 320 domain cells were used. A larger number of domains would have increased encoding times and provided marginally better compression ratios. The self-organizing method is faster than the filtered ranges method and therefore faster than the baseline method.

References

[1] Rajeshri, R. and Yashwant, S.C. (2013) Escape Time Fractals of Inverse Tangent Function. *International Journal of Computer and Organization Trends*, **3**, 16-21.

[2] Satyendra, K.P., Munshi, Y. and Arunima (2012) Fracint Formula for Overlaying Fractals. *Journal of Information Systems and Communication*, **3**, 347-352.

[3] Barnsley, M.F. and Sloan, A.D. (1987) Chaostic Compression. *Computer Graphics World*, **10**, 107-108.

[4] Barnsley, M. and Sloan, A. (1988) A Better Way to Compress Images. *Byte*, **13**, 215-223.

[5] Barnsley, M.F. (1988) Fractals Everywhere. Academic Press, New York.

[6] Jacquin, A.E. (1992) Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transaction on Image Processing*, **1**, 8-30. http://dx.doi.org/10.1109/83.128028

[7] Fisher, Y. (1994) Fractal Image Compression-Theory and Application. Springer-Verlag, New York.

[8] Sheridan, P. (1996) Spiral Architecture for Machine Vision. Ph.D. Thesis, University of Technology, Sydney.

[9] Bouboulis, P., Dalla, P.L. and Drakopoulos, V. (2006) Image Compression Using Recurrent Bivariate Fractal Interpolation Surfaces. *International Journal of Bifurcation and Chaos*, **16**, 2063-2071. http://dx.doi.org/10.1142/S0218127406015908

[10] Kramm, M. (2007) Compression of Image Clusters Using Karhunen Loeve Transformations. *Electronic Imaging, Human Vision*, **XII**, 101-106.

[11] Koli, N. and Ali, M. (2008) A Survey on Fractal Image Compression Key Issues. *Information Technology Journal*, 7, 1085-1095.

[12] Bressloff, P.C. and Stark, J. (1991) Neural Networks, Learning Automata and Iterated Function Systems. In: Crilly A.J., Earnshaw, R.A. and Jones, H., Eds., *Fractals and Chaos*, Springer-Verlag, 145-164. http://dx.doi.org/10.1007/978-1-4612-3034-2_8

[13] Jacquin, A.E. (1992) Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transaction on Image Processing*, **1**, 18-30. http://dx.doi.org/10.1109/83.128028

[14] Hamzaoui, R. (1995) Codebook Clustering by Self-Organizing Maps for Fractal Image Compression. *NATO ASI Conference Fractal Image Encoding and Analysis*, Trondheim, July 1995, 27-38.

[15] Kohonen, T. (1982) Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, **43**, 59-69. http://dx.doi.org/10.1007/BF00337288

[16] Krose, B. and der Smagt, P.V. (1996) An Introduction to Neural Networks. Amesterdam Univerisity, Amesterdam.

[17] Mandelbrot, B.B. (1982) The Fractal Geometry of Nature. Freeman Press, New York.

[18] Nikiel (2007) Iterated Function Systems for Real-Time Image Synthesis. Springer-Verlag, London.

[19] Al-Helali, B. (2010) Fractal Image Compression Using Iterated Function Systems. M.Sc Thesis, Taiz University, Yemen.

[20] Saupe, D., Hamzaoui, R. and Hartenstein, H. (1996) Fractal Image Compression—An Introductory Overview. In: Saupe, D. and Hart, J., Eds., *Fractal Models for Image Synthesis, Compression and Analysis*, ACM, New Orleans, SIGGRAPH'96 Course Notes 27.