Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2008, Article ID 426580, 15 pages
doi:10.1155/2008/426580

Research Article

Morphological Transform for Image Compression

Enrique Guzmán,[1] Oleksiy Pogrebnyak,[2] Cornelio Yañez,[2] and Luis Pastor Sanchez Fernandez[2]

[1] *Universidad Tecnológica de la Mixteca, Carretera Acatlima km 2.5, Huajauapan de León, CP 69000, Oaxaca, Mexico*
[2] *Centro de Investigación en Computación, Instituto Politécnico Nacional, Ave. Juan de Dios Batiz S/N, esq. Miguel Othon de Mendizabal, CP 07738, Mexico*

Correspondence should be addressed to Oleksiy Pogrebnyak, olek@pollux.cic.ipn.mx

Received 29 August 2007; Revised 30 November 2007; Accepted 4 April 2008

Recommended by Sébastien Lefèvre

A new method for image compression based on morphological associative memories (MAMs) is presented. We used the MAM to implement a new image transform and applied it at the transformation stage of image coding, thereby replacing such traditional methods as the discrete cosine transform or the discrete wavelet transform. Autoassociative and heteroassociative MAMs can be considered as a subclass of morphological neural networks. The morphological transform (**MT**) presented in this paper generates heteroassociative MAMs derived from image subblocks. The **MT** is applied to individual blocks of the image using some transformation matrix as an input pattern. Depending on this matrix, the image takes a morphological representation, which is used to perform the data compression at the next stages. With respect to traditional methods, the main advantage offered by the **MT** is the processing speed, whereas the compression rate and the signal-to-noise ratio are competitive to conventional transforms.

1. INTRODUCTION

The data transformation stage of image coding facilitates information compression at the further stages. Its purpose is twofold: transform the image pixels into a representation where they are noncorrelated and identify the less important parts of the image isolating various spatial frequencies of the image. Although a great variety of existing transformations can be employed in image compression, only two of them are actually used to this end: the discrete cosine transform (DCT) and the discrete wavelet transform (DWT).

The DCT was proposed by Ahmed et al. [1]. Ever since, diverse algorithms of a DCT implementation have been developed to achieve the least possible computational complexity. Chen et al. [2] proposed one of the first algorithms for a fast DCT implementation. This algorithm takes advantage of the cosine symmetry function thus reducing the number of operations necessary for DCT calculation. Arai et al. [3] suggested a more efficient and fast variant of a fast DCT scheme applied to images. This algorithm uses only the real part of the discrete Fourier transform [4], and the coefficients are calculated with the help of the fast Fourier transform algorithm described by Winograd [5]. Actually, DCT is used in JPEG image compression and MPEG video compression standards [6, 7].

DeVore et al. [8] developed a mathematical theory enabling to use the wavelet transform in image compression. Daubechies and her collaborators proposed a scheme for image compression with the help of the DWT. The scheme employs biorthogonal filters to obtain a set of image subbands using a pyramidal architecture algorithm. This decomposition provides the subband images corresponding to different levels of resolution and orientation [9]. Lewis and Knowles [10] proposed a scheme for image compression based on 2D wavelet transform to separate the image by its spatial elements and spectral coefficients.

Various methods for coding of image wavelet coefficients are known. The first wavelet image coding algorithm of embedded zerotree wavelet (EZW) was proposed by Shapiro [11]. Next, Said and Pearlman [12] proposed a new and better implementation of the EZW, the algorithm of set partitioning in hierarchical trees (SPIHTs). It is based on the use of data sets organized in hierarchical trees. A new algorithm for image compression known as embedded block coding with optimized truncation (EBCOT) was proposed by Taubman in 2000 [13]. In this algorithm, each subband

is divided into small blocks of wavelet coefficients called "blocks code," and then chains of bits separated for each block code are generated. These chains can be truncated independently to different lengths. The JPEG2000 image compression standard is based fundamentally on the DWT and EBCOT [14].

On the other hand, a new technology for image compression based on artificial neuronal networks (ANNs) has arisen as an alternative to traditional methods. Within the novel approach, new image compression schemes were created and the existing algorithms were essentially modified.

The selforganizing map (SOM) ANN has been used with a great deal of success in creating codebooks for vector quantization (VQ). The SOM is a competitive-learning network; it was developed by Professor Kohonen in the early 1980s [15, 16]. One of the first works where SOMs were used for image compression was presented by Bogdan and Meadows [17]. Their algorithm is based on the use of the SOMs and fractal coding to find similar features in different resolution representations of the image. In this process, patterns are mapped onto the two-dimensional array of formal neurons forming a codebook similar to VQ coding. The SOM ordering properties allow finding not only the mapping of the best feature match neuron but also its neighbors in the network. This modification reduced the computational load when finding and removing redundancies between scale representations of the original image. Amerijckx et al. [18] proposed a lossy compression scheme for digital still images using Kohonen's neural network algorithm. They applied the SOM at both quantification and codification stages of the image compressor. At the quantification stage, the SOM algorithm creates a correspondence between the input space of stimuli, and the output space constituted of the codebook elements (codewords, or neurons) derived using the Euclidean distance. After learning the network, these codebook elements approximate the vectors in the input space in the best possible way. At the entropy coder stage, a differential entropy coder uses the topology-preserving property of the SOMs resulted from the learning process and the hypothesis that the consecutive blocks in the image are often similar. In [19], the same authors proposed an image compression scheme for lossless compression using SOMs and the same principles. Mokhtari and Boukelif [20] presented a new algorithm based on Kohonen's neural network, which accelerates the fractal image compression. Kohonen's network is used in an adaptive algorithm that searches the best range block for a source block with the affine transformation and both contrast and brightness parameters. When the difference between blocks is higher than a predefined threshold, the source block is subdivided into four subblocks. This division keeps repeating until either the difference is lower than the threshold or the minimal block size is reached. The main disadvantage of SOM algorithms is that a long training time is required due to the fact that the network starts with random initial weights. Panchanathan et al. [21] used the backward error propagation algorithm (BEP) to rapidly obtain the initial weights, which are then used to speed up the training time required by the SOFM algorithm. The proposed approach (BEP-SOFM) combines the advantages of both techniques and, hence, achieves a good coding performance in a shorter training time.

Another type of ANN that has been used widely in image compression is the feedforward network. It is classified in the category of *signal-transfer* network, and its learning process is defined by the error backpropagation algorithm. Setiono and Lu [22] described the feedforward neural network algorithm applied to image compression. The neural network construction algorithm begins with a simple network topology containing a single unit in the hidden layer. An optimal set of weights for this network is obtained applying a variant of the quasi-Newton method for unconstrained optimization. If this set of weights does not give a network with the desired accuracy, then one more unit is added to the hidden layer, and the network is retrained. The process is repeated until the desired network is obtained. This algorithm has a longer training time but with each addition of the hidden unit to the network the signal-to-noise ratio of the compressed image is increased. In [23], a linear selforganized feedforward neural network for image compression is presented. The first step in the coding process is to divide the image into square blocks of size $m \times m$, each block represents a feature vector of m^2 dimension in the feature space. Then, a neural network of the input dimension of m^2 and output dimension of m extracts the principal components of the autocorrelation matrix of the input image using the generalized Hebbian learning algorithm (GHA). Training based on GHA for each block then yields a weight matrix of $m \times m^2$ size. Its rows are the eigenvectors of the autocorrelation matrix of the input image block. Projection of each image block onto the extracted eigenvectors yields m coefficients for each block. Then image compression is accomplished quantizing and coding the coefficients for each block. Roy et al. [24] developed the image compression technique that preserves edges using one hidden layer feedforward neural network. Its neurons are determined adaptively based on the image to be compressed. First, in order to reduce the size considerably, several image processing steps, namely, edge detection, thresholding, and thinning, are applied to the image. The main concern of the second phase is to determine adaptively the structure of the NN that encodes the image using backpropagation training method: the processed image block is fed as a single input pattern while the single output pattern has been constructed from the original image. Furthermore, this method proposes the initialization of the weights between the input layer and the lone hidden layer transforming pixel coordinates of the input pattern block into its equivalent one-dimensional representation. This initialization process exhibits a better rate of convergence of the backpropagation training algorithm in comparison to the randomization of the initial weights.

The following examples show a direct relationship between the ANN methods and the methods based on DCT and DWT. In [25] Ng and Cheng proposed the implementation of the DCT with ANN structures. The structured artificial neural network is placed in four major subnetworks: one for forward DCT (back propagation NN of $64 \times 16 \times 63$), one for energy classification (back

propagation NN of $63 \times 32 \times 4$), one for inverse DCT (back propagation NN of $63 \times 16 \times 64$) and one for direct current (DC) adjustment (back propagation NN of $64 \times 2 \times 1$). Each subnetwork is trained and tested individually and independently except the DC adjustment network. On the other hand, Burges et al. [26] used a nonlinear predictor, implemented with ANN, to predict wavelet coefficients for image compression. The process consists of reducing the variance of the residual coefficients; then, the nonlinear predictor can be used to reduce the compressed bitstream length. In order to implement the neural network predictor, the authors considered two-layer neural network with the single output parameterized by the vector of weights. The output unit is a sigmoid, taking values in $[0, 1]$. The network is trained for each subband and each wavelet level, and the outputs are translated and rescaled, again per each subband and wavelet level. Similarly, the inputs are rescaled so their values mostly lie in the interval $[-1, 1]$. The mean-squared error measure is used to train the net in order to minimize the variance of the prediction residuals.

Two interesting proposals of ANN application to image compression must be mentioned. First of them describes a practical and effective image compression system based on multilayer neural network [27]. The suggested system consists of two multilayer neural networks that compress the image in two stages. The first network compresses the image itself, and the second one compresses the difference between the reconstructed and the original images. In the second proposal, Danchenko et al. [28] developed a program for compression of color and grayscale images using ANN. This program was named the neural network image compressor (NNIC). The NNIC implements two image compression methods based on multilayer perceptron and Kohonen neural network architectures. Finally, an algorithm based on the DCT complements to the NNIC program.

Ritter et al. [29] introduced the concept of a morphological neural network. They proposed to compute the total input effect on the ith neuron with the help of the dilation and erosion operations of mathematical morphology. Then, in 1996, Ritter and Sussner [30] proposed morphological associative memories (MAMs) on the base of the morphological neural networks. Two years after, Ritter et al. [31] extensively developed the concept of MAMs. In this paper, we present a new image transform applied to image compression based on MAMs. For image compression purposes, we used heteroassociative MAMs of minimum type at the transformation stage of image coding instead of DCT or DWT. This way, the morphological transform for image compression was derived.

We will also mention an interesting work done by Sussner and Valle [32]. The gist of this paper is that the authors characterize the fixed points and basins of attraction of grayscale AMMs in order to derive rigorous mathematical results on the storage capacity and the noise tolerance of these memories. Moreover, a modified model with improved noise tolerance is presented and AMMs are successfully used for pattern classification.

The paper is organized as follows. In Section 2, a brief theoretical background of MAM is given. Section 3 describes the proposed **MT** algorithm. Numerical simulation results obtained for the conventional image compression techniques and the **MT** are provided and discussed in Section 4. Finally, conclusions are given in Section 5.

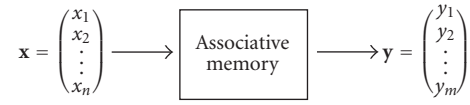

Figure 1: Associative memory scheme.

2. THEORETICAL BACKGROUND OF MORPHOLOGICAL ASSOCIATIVE MEMORIES

The modern era of associative memories began in 1982 when Hopfield developed the Hopfield's associative memory [33]. Hopfield's work recovered the investigators' interest in such areas as artificial neuronal networks and associative memories forgotten until that moment.

The associative memory is an element whose fundamental intention is to recover patterns even if they contain dilative, erosive or random noise. The generic associative memory scheme is shown in Figure 1. The input patterns and output patters are represented by $\mathbf{x}$ and $\mathbf{y}$, respectively; n and m are integer positive numbers that represent the dimensions of the input and output patterns.

Let $\{(\mathbf{x}^1,\mathbf{y}^1),(\mathbf{x}^2,\mathbf{y}^2),\ldots,(\mathbf{x}^k,\mathbf{y}^k)\}$ be k vector pairs defined as the *fundamental set of associations*. The fundamental set of associations is represented by

$$\{(\mathbf{x}^\mu,\mathbf{y}^\mu) \mid \mu = 1,2,\ldots,k\}. \tag{1}$$

The associative memory is represented by a matrix and is generated from the fundamental set of associations.

2.1. Morphological associative memories

The MAMs base their functioning on the morphological operations, dilation and erosion [34]. This results that MAMs use the maximums or minimums of sums [31]. This feature distinguishes them from the Hopfield's memories, which use sums of products.

One can define the operations necessary for the learning process of MAM and the recovery process when the fundamental set is delineated. These operations use the calculation of the binary operations of maximum $\vee$ and minimum $\wedge$ [31].

Let $\mathbf{d}$ be a column vector of dimension m, and let $\mathbf{f}$ be a row vector of dimension n, then the *maximum product* is given by

$$\mathbf{d} \nabla \mathbf{f} = \mathbf{C} = [c_{ij}]_{m\times n}, \tag{2}$$

where $c_{ij} = (d_i + f_j)$.

Generalizing for a fundamental set of associations,

$$c_{ij} = \bigvee_{l=1}^{k} (d_{il} + f_{lj}). \tag{3}$$

The *minimum product* is given by

$$\mathbf{d}\Delta\mathbf{f} = \mathbf{C} = [c_{ij}]_{m\times n}. \tag{4}$$

For a fundamental set of associations, c_{ij} is defined by

$$c_{ij} = \bigwedge_{l=1}^{k} (d_{il} + f_{lj}). \tag{5}$$

On the other hand, let $\mathbf{D} = [d_{ij}]_{m\times n}$ be a matrix and $\mathbf{f} = [f_i]_n$ a column vector, the calculation of the maximum product $\mathbf{D}\nabla\mathbf{f}$ results in a column vector $\mathbf{c} = [c_i]_n$, where c_i is defined by

$$c_i = \bigvee_{j=1}^{n} (d_{ij} + f_j). \tag{6}$$

For the minimum product $\mathbf{c} = \mathbf{D}\Delta\mathbf{f}$,

$$c_i = \bigwedge_{j=1}^{n} (d_{ij} + f_j). \tag{7}$$

According to the mode of operation, the MAMs are classified in two groups:

(i) autoassociative morphological memories,

(ii) heteroassociative morphological memories (HHMs).

From (2) to (7) are used in the MAMs of both heteroassociative and autoassociative operation modes. Due to certain characteristics required by image compression application discussed later, HMMs are of particular interest.

A morphological associative memory is heteroassociative if $\exists\mu \in \{1,2,\ldots,k\}$ such that $\mathbf{x}^\mu \neq \mathbf{y}^\mu$. There are two types of morphological heteroassociative memories: HMM *max*, symbolized by $\mathbf{M}$, and HMM *min*, symbolized by $\mathbf{W}$.

2.1.1. Morphological heteroassociative memories min

The HMMs *min* ($\mathbf{W}$) are those that use the maximum product (2) and the minimum operator $\wedge$ in their learning phase and the maximum product in their recovery phase.

Learning phase:

(1) the matrices $\mathbf{y}^\mu\nabla(-\mathbf{x}^\mu)^t$ are calculated for each k element of the fundamental set of associations $(\mathbf{x}^\mu, \mathbf{y}^\mu)$,

(2) the memory $\mathbf{W}$ is obtained applying the minimum operator $\wedge$ to the matrices resulted from step (1). $\mathbf{W}$ is given by

$$\begin{aligned} \mathbf{W} &= \bigwedge_{\mu=1}^{k} \left[\mathbf{y}^\mu\nabla\left(-\mathbf{x}^\mu\right)^t\right] = [w_{ij}]_{m\times n}, \\ w_{ij} &= \bigwedge_{\mu=1}^{k} (y_i^\mu - x_j^\mu). \end{aligned} \tag{8}$$

Recovery phase:

(1) the maximum product $\mathbf{W}\nabla\mathbf{x}^\omega$, where $\omega \in \{1,2,\ldots,k\}$, is calculated. Then the column vector $\mathbf{y} = [y_i]_m$, which represents the output patterns associated with $\mathbf{x}^\omega$ input patterns, is obtained as

$$\begin{aligned} y &= \mathbf{W}\nabla\mathbf{x}^\omega, \\ y_i &= \bigvee_{j=1}^{n} (w_{ij} + x_j^\omega). \end{aligned} \tag{9}$$

The following theorem and corollary from [31] govern the conditions that must be satisfied by HMM *min* to obtain a perfect recall to output patterns. Here we reproduce them.

Theorem 1 (see [31, Theorem 2]). *$\mathbf{W}\nabla\mathbf{x}^\omega = \mathbf{y}^\omega$ for all $\omega = 1,\ldots,k$ if and only if for each ω and each row index $i = 1,\ldots,m$ there are column indexes $j_i^\omega \in \{1,\ldots,n\}$ such that $m_{ij_i^\omega} = y_i^\omega - x_{j_i^\omega}^\omega$ for all $\omega = 1,\ldots,k$.*

Corollary 1 (see [31, Corollary 2.1]). *$\mathbf{W}\nabla\mathbf{x}^\omega = \mathbf{y}^\omega$ for all $\omega = 1,\ldots,k$ if and only if for each row index $i = 1,\ldots,m$ and each $\gamma \in \{1,\ldots,k\}$ there is a column index $j_i^\gamma \in \{1,\ldots,n\}$ such that*

$$x_{j_i^\gamma}^\gamma = \bigvee_{\varepsilon=1}^{k} \left(x_{j_i^\gamma}^\varepsilon - y_i^\varepsilon\right) + y_i^\gamma. \tag{10}$$

On the other hand, the following theorem indicates the amount of noise permissible in the input patterns to obtain a perfect recall to output patterns.

Theorem 2 (see [31, Theorem 3]). *For $\gamma = 1,\ldots,k$, let $\tilde{\mathbf{x}}^\gamma$ be a corrupted input pattern of $\mathbf{x}^\gamma$. Then $\mathbf{W}\nabla\tilde{\mathbf{x}}^\gamma = \mathbf{y}^\gamma$ if and only if it satisfies that*

$$\tilde{x}_j^\gamma \leq x_j^\gamma V \bigwedge_{i=1}^{m} \left(\bigvee_{\varepsilon\neq\gamma} [y_i^\gamma - y_i^\varepsilon + x_i^\varepsilon]\right) \quad \forall\, j = 1,\ldots,n, \tag{11}$$

and for each row index $i \in \{1,\ldots,m\}$ there is a column index $j_i \in \{1,\ldots,n\}$ such that

$$\tilde{x}_{ji}^\gamma = x_{ji}^\gamma V \left(\bigvee_{\varepsilon\neq\gamma} [y_i^\gamma - y_i^\varepsilon + x_{ji}^\varepsilon]\right). \tag{12}$$

3. MORPHOLOGICAL TRANSFORM USING MAM

The data transformation stage in a system for image codification has the aim of facilitating information compression in the later stages. The **MT** is proposed as an alternative to traditional transformation methods. The algorithm of this model uses the MAMs to generate a morphological representation of an image. As it was mentioned above, the MAMs are based on the morphological operations that calculate the maximums or minimums of sums. This feature makes MAM to be a model with a high-processing speed, and the **MT** inherits this property.

The following features make **MT** attractive to be used in the transformation stage within an image compression

system:

(i) the morphological representation of the image, generated by the **MT**, can facilitate information compression in the following stages;

(ii) the **MT** is reversible;

(iii) in the image transformation process, the **MT** has low-memory requirements (space complexity). It uses a limited arithmetical precision, and is implemented with a few basic arithmetical operations (has low-time complexity).

The MAMs have turned out to be an excellent tool for recognizing and recovering patterns, even if the patterns contain dilative, erosive, or random noise [31]. At the inverse **MT** stage, this feature allows to suppress some of noise generated at other image compression stages.

As it was mentioned above, MAM can be autoassociative or heteroassociative. A morphological associative memory is autoassociative if $\mathbf{x}^{\mu} = \mathbf{y}^{\mu}, \mu \in \{1, 2, \ldots, k\}$. This fact discards the use of autoassociative morphological memories in the **MT** algorithm because the image to be compressed would not be available in the decompression process to perform the inverse **MT** process.

A heteroassociative associative memory allows associating input patterns with different output patterns in content and dimension. Taking this property into account, HMM can be used in the **MT** algorithm, where the image will be sectioned to form output patterns, and input patterns will be predefined as a transformation matrix. The transformation matrix will be available in both compression and decompression processes thus allowing to implement the inverse morphological transformation (**IMT**). The HMM used in the **MT** can be of *min* or *max* type. That is why the **MT** is immune to erosive or dilative noise, respectively.

3.1. Preliminary definitions

The proposed **MT** is applied to individual blocks of the image. Let the image be represented by a matrix, $\mathbf{A} = [a_{ij}]_{m \times n}$, where m is the image height and n is the image width; and a represents the ijth pixel value: $a \in \{0, 1, 2, \ldots, 2^L - 1\}$, where L is the number of bits necessary to represent the value of a pixel.

The **MT** presented in this paper generates heteroassociative MAMs derived from image subblocks. Next, we define the *image subblock* and *image vector* terms.

Definition 1 (image subblock (**sb**)). Let $\mathbf{A} = [a_{ij}]$ be an $m \times n$ matrix representing an image, and let $\mathbf{sb} = [\mathrm{sb}_{ij}]$ be a $d \times d$ matrix. The **sb** matrix is defined as a subblock of the **A** matrix if the **sb** matrix is a subgroup of the **A** matrix such that

$$\mathrm{sb}_{ij} = a_{\delta_i \tau_j}, \tag{13}$$

where $i, j = 1, 2, 3, \ldots, d$, $\delta = 1, 2, 3, \ldots, m$, $\tau = 1, 2, 3, \ldots, n$ and $a_{\delta_i \tau_j}$ represents the value of the pixel determined by the coordinates $(\delta + i, \tau + j)$, where (δ, τ) and $(\delta + d, \tau + d)$ are the beginning and the end of the subblock,respectively.

Definition 2 (image vector (**vi**)). Let $\mathbf{sb} = [\mathrm{sb}_{ij}]$ be an image subblock and let $\mathbf{vi} = [\mathrm{vi}_i]$ be a vector of size d. The ith row of the **sb** matrix is said to be an *image vector* **vi** such that

$$\mathrm{vi}_i = [\mathrm{sb}_{i1}, \mathrm{sb}_{i2}, \ldots, \mathrm{sb}_{id}], \tag{14}$$

where $i = 1, 2, 3, \ldots, d$. From each *image subblock*, d *image vectors* can be obtained:

$$\mathbf{vi} = [\mathrm{sb}_{\mu 1}, \mathrm{sb}_{\mu 2}, \ldots, \mathrm{sb}_{\mu d}], \tag{15}$$

where $\mu = 1, 2, 3, \ldots, d$.

The **MT** uses a *transformation matrix*, which is formed by transformation vectors. These two terms are defined below.

Definition 3 (transformation vectors (**vt**)). Let $\mathbf{vt} = [\mathrm{vt}_i]$ be a vector of size d. The **vt** vector is called a *transformation vector* when it is used in both processes of MAM learning and pattern recovery, whose generation is governed by [31, Theorem 2 and Corollary 2.1].

Definition 4 (transformation matrix (**mt**)). Let $\mathbf{vt} = [\mathrm{vt}_i]_d$ be a transformation vector. The set formed by d transformation vectors $\{\mathbf{vt}^1, \mathbf{vt}^2, \ldots, \mathbf{vt}^d\}$ is called *transformation matrix* $\mathbf{mt} = [\mathrm{mt}_{ij}]_{d \times d}$, where the ith row o matrix **mt** is represented by vector $\mathbf{vt}^i$. Then the ijth component of **mt** is defined by

$$\mathrm{mt}_{ij} = \mathrm{vt}_j^i \mid i, j = 1, 2, \ldots, d. \tag{16}$$

3.2. Morphological transform using HMM min

The matrix **A** is divided into $N = (m/d) \cdot (n/d)$ submatrices, or image subblocks of $d \times d$ size, each of them is divided into d image vectors of d size: $\mathbf{vi}^{\mu} = [\mathrm{vi}_i]_d \mid \mu = 1, 2, \ldots, d$.

The **MT** process generates N MAMs, structured in a matrix form to represent the morphological transformation

$$\mathbf{MT} = \mathbf{O}\{\mathrm{MAM}^{ij}\} = \begin{pmatrix} \mathrm{MAM}^{11} & \mathrm{MAM}^{12} & \cdots & \mathrm{MAM}^{1\eta} \\ \mathrm{MAM}^{21} & \mathrm{MAM}^{22} & \cdots & \mathrm{MAM}^{2\eta} \\ \vdots & \vdots & & \vdots \\ \mathrm{MAM}^{\lambda 1} & \mathrm{MAM}^{\lambda 2} & \cdots & \mathrm{MAM}^{\lambda\eta} \end{pmatrix}, \tag{17}$$

where $i = 1, 2, \ldots, \lambda$, $j = 1, 2, \ldots, \eta$, $\lambda = m/d$, and $\eta = n/d$; in addition, operator $\mathbf{O}\{\cdot\}$ is defined to represent such an organization where MAMs constitute the transformation matrix **MT**. Thus, the MAM^{ij} represents the generated memory when MAM learning process is applied to the ijth image subblock.

When an HMM *min* is used in order to transform an image subblock of $d \times d$ size, the **MT** is defined by the

following expression:

$$\begin{aligned}\mathbf{MT}_{\min} &= \mathbf{O}\{\mathrm{MAM}_{\min}^{xy} \mid x = 1,2,\ldots,\lambda,\ y = 1,2,\ldots,\eta\},\\ \mathrm{MAM}_{\min}^{xy} &= \bigwedge_{\mu=1}^{d}\left[\mathbf{vi}^{\omega_\mu}\nabla\left(-\mathbf{vt}^{\mu}\right)^{T}\right]\\ &= \left[w_{ij}\right]_{d\times d}^{xy} \mid \omega = 1,2,\ldots,N,\\ \left[w_{ij}\right]_{d\times d}^{xy} &= \bigwedge_{\mu=1}^{d}\left(\mathrm{vi}_i^{\omega_\mu} - \mathrm{vt}_j^{\mu}\right) \mid i,j = 1,2,\ldots,d,\end{aligned} \tag{18}$$

where ω indicates to what N image subblock the image vectors belong; thus, $\mathbf{vi}^{\omega_\mu}$ is the μth row of the ωth image subblock.

The **vt** vectors form transformation matrix $\mathbf{mt} = [\mathrm{mt}_i]_d$. It affects the resulted parameters such as the compression ratio and the signal-to-noise ratio. The transformation matrix must be known at both image coding and image decoding stages.

There exist a great variety of values that satisfy the conditions governing the generation of the transformation matrix. As an option, one can choose the elements of transformation vectors under the following conditions:

$$\mathrm{vt}_{n'}^{m''}\begin{cases} = 0 & m' \neq n',\\ > e & m' = n',\end{cases} \tag{19}$$

where e is the maximum value that can take an element of the image **A**.

As a result of applying the **MT** to the image, N associative memories **W** of size $d \times d$ are obtained. This set of memories forms the transformed image. Figure 2 shows the **MT** scheme that uses HMMs. The image information remains concentrated within minimum values. Thus, it is possible to obtain some advantages of this new image representation at the next stages of image coding. Figure 3 shows **MT** results on byte represented grayscale images of 512×512 size.

The inverse process, the *inverse morphological transform* (**IMT**), consists of applying the recovery phase of an HMM between the transformation vectors and each HMM that forms the **MT**.

As a result of the **IMT** process, N image subblocks are generated, which altogether represent the original image transformed by the **MT**:

$$\mathbf{IMT} = \mathbf{O}\{\mathbf{sb}^{ij}\} = \begin{pmatrix} \mathrm{sb}^{11} & \mathrm{sb}^{12} & \cdots & \mathrm{sb}^{1\eta}\\ \mathrm{sb}^{21} & \mathrm{sb}^{22} & \cdots & \mathrm{sb}^{2\eta}\\ \vdots & \vdots & & \vdots\\ \mathrm{sb}^{\lambda 1} & \mathrm{sb}^{\lambda 2} & \cdots & \mathrm{sb}^{\lambda\eta}\end{pmatrix}, \tag{20}$$

where $i = 1,2,\ldots,\lambda$, $j = 1,2,\ldots,\eta$, $\lambda = m/d$, and $\eta = n/d$. The operator $\mathbf{O}\{\cdot\}$ is used because the matrices **sb** within the **IMT** keep the same position that the MAMs used for their recovery keep within the **MT**.

The **IMT** is possible because

(i) the transformed image is an HMM set,

(ii) the transformation matrix is available at the decompression stage.

For an **IMT** process, two cases can be defined.

Case 1 (when the **MT** has not been altered by noise). This is a reversible, lossless process. Nevertheless, the obtained compression ratio is not significant.

When an HMM *min* was used in order to transform an image subblock of $d \times d$ size, the **IMT** is defined by the following expression:

$$\begin{aligned}\mathbf{IMT}_{\min} &= \mathbf{O}\{\mathbf{sb}^{xy} \mid x = 1,2,\ldots,\lambda,\ y = 1,2,\ldots,\eta\},\\ \mathbf{sb}^{xy} &= \mathbf{vi}^{(xy)_\mu} \mid \mu = 1,2,\ldots,d,\\ \mathbf{vi}^{(xy)_\mu} &= \mathrm{HMM}_{\min}^{xy}\nabla\mathbf{vt}^{\mu} = \left[\mathrm{vi}_i^{(xy)_\mu}\right]_d,\\ \mathrm{vi}_i^{(xy)_\mu} &= \bigvee_{j=1}^{d}\left(w_{ij}^{xy} + \mathrm{vt}_j^{\mu}\right),\end{aligned} \tag{21}$$

where xy indicates to what N image subblock the image vectors belong; thus, $\mathbf{vi}^{(xy)_\mu}$ is the μth row of the xyth image subblock.

Case 2 (when the **MT** has been altered by noise). This is an irreversible process, the recovered image is an altered version of the original image. Nevertheless, the obtained compression ratio is significant.

The next stage of image coding is the quantization. This stage modifies the **MT** information. **MT** is a set of HMM, and the theory of MAMs presented in [31] only considers a perfect recall to output patterns when the noise appears in the input patterns and not in the associative memories. If the modification of the information contained in the obtained memories **W** at **MT** process is considered as noise, then, how does this noise affect the associative memory in the recovery of the original output patterns (blocks of the original image)?

In order to answer this question, we formulated a new theorem in MAM theory [35].

Theorem 3. *Let* $\widetilde{\mathbf{W}}$ *denote the distorted version of the associative memory* **W***:*

$$\begin{aligned}\widetilde{W} &= W \pm r,\\ \widetilde{w}_{ij} &= w_{ij} \pm r,\end{aligned} \tag{22}$$

where r *represents the noise associated with* **W**. *Then*

$$\widehat{\mathbf{W}}\nabla\mathbf{x}^{\gamma} = \widehat{\mathbf{y}}^{\gamma} = y_i^{\gamma} \pm r. \tag{23}$$

Proof. Considering the theorem [31, Theorem 2] and its respective corollary [31, Corollary 2.1], we have $\mathbf{y}^{\gamma} = \mathbf{W}\nabla\mathbf{x}^{\gamma}$, bearing in mind the corrupted version of the associative

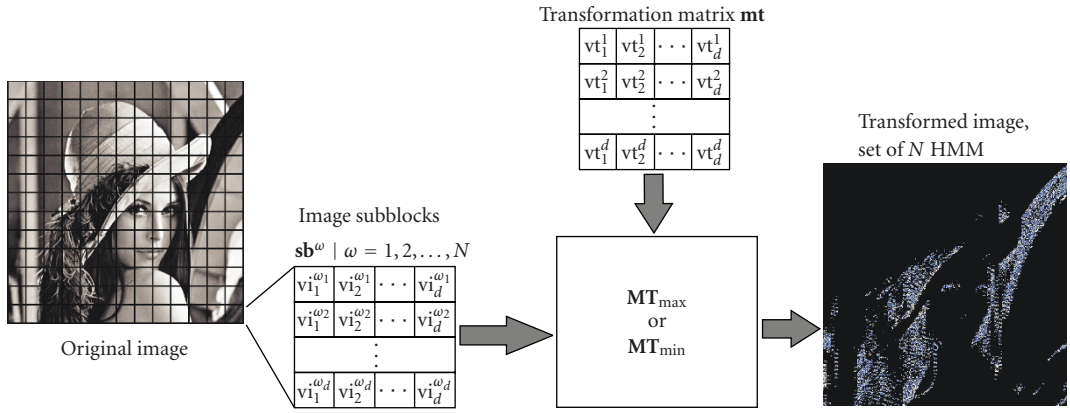

Figure 2: **MT** scheme using HMMs.

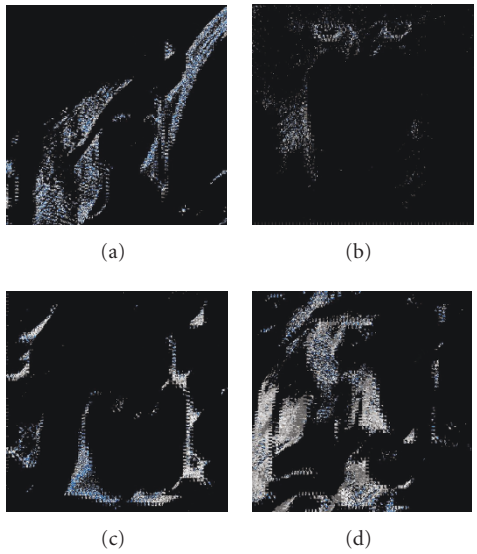

Figure 3: **MT** results on (a) Lena, (b) Baboon, (c) Peppers, (d) Man.

memory, then $\mathbf{y}^{\gamma} = \bullet \nabla \mathbf{x}^{\gamma}$:

$$\begin{aligned}
\left(\mathbf{W} \nabla \mathbf{x}^{\gamma}\right)_t &= \bigvee_{j=1}^{n} \left(\widetilde{w}_{ij} + x_j^{\gamma}\right), \\
&\geq \widetilde{w}_{ij_i} + x_{j_i}^{\gamma} \\
&= \widetilde{w}_{ij_i} + \left(\bigvee_{\varepsilon=1}^{k} \left(x_{j_i}^{\varepsilon} - y_i^{\varepsilon}\right) + y_i^{\gamma} \right) \\
&= \widetilde{w}_{ij_i} + y_i^{\gamma} - \bigwedge_{\varepsilon=1}^{k} \left(y_i^{\varepsilon} - x_{j_i}^{\varepsilon}\right) \\
&= \widetilde{w}_{ij_i} + y_i^{\gamma} - w_{ij_i} \\
&= w_{ij_i} \pm r + y_i^{\gamma} - w_{ij_i} \\
&= y_i^{\gamma} \pm r.
\end{aligned} \tag{24}$$

Theorem 3 shows that the noise r associated with the associative memory directly affects the output patterns and the property of the image perfect recovery. The noise r associated with the set of associative memories depends directly on the used quantification factor.

Considering Theorem 3, expression (3) is rewritten to define the **IMT** for Case 2

$$\begin{aligned}
\mathbf{IMT}_{\min} &= \mathbf{O}\{\mathbf{sb}^{xy} \mid x = 1, 2, \ldots, \lambda,\ y = 1, 2, \ldots, \eta\}, \\
\mathbf{sb}^{xy} &= \mathbf{vi}^{(xy)_{\mu}} \mid \mu = 1, 2, \ldots, d, \\
\mathbf{vi}^{(xy)_{\mu}} &= \mathrm{HMM}_{\min}^{xy} \nabla \mathbf{vt}^{\mu} = \left[\mathrm{vi}_i^{(xy)_{\mu}}\right]_d \\
\mathrm{vi}_i^{(xy)_{\mu}} &= \bigvee_{j=1}^{d} \left(w_{ij}^{xy} + \mathrm{vt}_j^{\mu} \pm r\right).
\end{aligned} \tag{25}$$

The **IMT** scheme using HMM is shown in Figure 4. □

3.3. Complexity of MT algorithm

The algorithm complexity is measured by two parameters: the *time* complexity, or how many steps it needs, and the *space* complexity, or how much memory it requires. In this subsection, we analyze time and space complexity of the **MT** algorithm. For this purpose, we will use pseudocode of the most significant part of the presented **MT** algorithm shown in Algorithm 1.

3.3.1. Time complexity

In order to measure the **MT** algorithm time complexity, we first obtain the run time based on the number of elementary operations (EOs) that **MT** realizes to calculate one image subblock. This calculation is the most representative element of the **MT** algorithm.

Considering pseudocode from Algorithm 1, one can conclude that in the *worst case*, the condition of line 9 will always be true. Therefore, line 10 will be executed in all

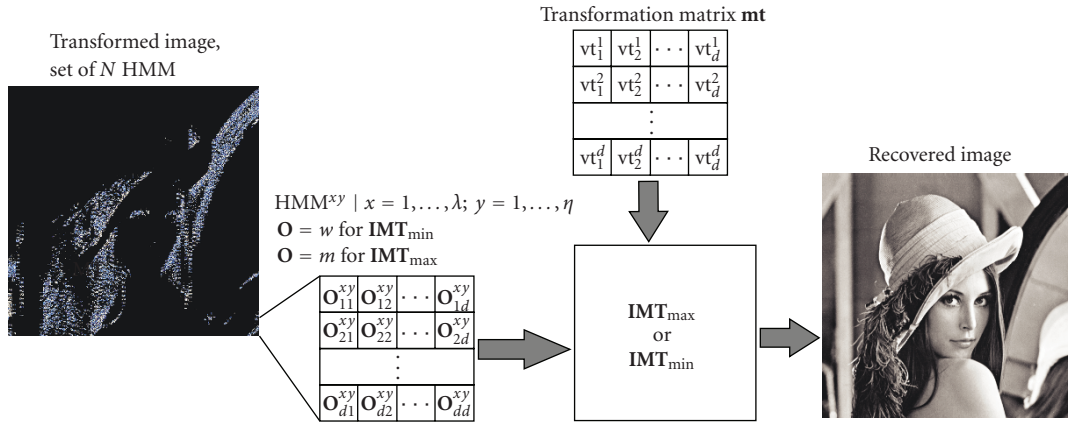

Figure 4: **IMT** scheme using HMMs.

```
01| subroutine P_min()
02| variables
03|    y, x, l, aux: integer
04| begin
05|    for l ← 0 to k [operations l = l + 1] do
06|       for y ← 0 to d [operations y = y + 1] do
07|          for x ← 0 to d [operations x = x + 1] do
08|             aux = vi[l + y] − vt[l + x]:
09|             if (aux < w[x][y]) then
10|                w[x][y] = aux;
11|             and_if
12|          end_for
13|       end_for
14|    end_for
15| end_subroutine
```

Algorithm 1: Pseudocode of the algorithm for HMM *min* computation.

iterations, and then the internal loop realizes the following number of EO:

$$\left(\sum_{x=0}^{d}(10+3)\right)+3=13\left(\sum_{x=0}^{d}1\right)+3=13d+3. \tag{26}$$

The next loop will repeat $13(d)+3$ EO at each iteration:

$$\left(\sum_{y=0}^{d}(13d+3)+3\right)+3=\left(\sum_{y=0}^{d}13d+6\right)+3$$
$$=d(13d+6)+3=13d^2+6d+3. \tag{27}$$

The last loop will repeat the same number of EO at each iteration. Also, this loop will be repeated k times, where k represents the number of elements of the fundamental set of associations. Thus, the total number of EO that realizes the algorithm is

$$T(n)=k\left(13d^2+6d+6\right)+3. \tag{28}$$

Based on expression (28), we can conclude that the order of growth of the proposed algorithm is $\mathbf{O}(n^2)$.

3.3.2. Space complexity

The **MT** algorithm space complexity is determined by the amount of memory required for its execution.

The transformation process of $d \times d$ image subblock requires two vectors, $\mathbf{vt}[d\times d]\mathbf{vi}[d\times d]$, and a matrix $\mathbf{w}[d][d]$. Hence, the number of memory units required for this process is

$$\text{un\_P1} = \text{un\_}\mathbf{vt} + \text{un\_}\mathbf{vi} + \text{un\_}\mathbf{w}. \tag{29}$$

The transformed image needs for its storage the matrix MT $[h\_i][w\_i]$, where h_i is the image height and w_i is the image width. The number of memory units required for this process is

$$\text{un\_P2} = \text{un\_MT}. \tag{30}$$

The total number of memory units required by the **MT** algorithm is the sum of the units required by the P1 and P2 processes:

$$\begin{aligned}\text{un\_P1} + \text{un\_P2} &= \text{un\_}\mathbf{vt} + \text{un\_}\mathbf{vi} + \text{un\_}\mathbf{w} + \text{un\_MT}\\ &= (d)(d) + (d)(d) + (d)(d) + (h\_i)(w\_i)\\ &= 3d^2 + (h\_i)(w\_i).\end{aligned} \tag{31}$$

The **MT** algorithm uses only summation, subtraction, and comparison operations. Therefore, the result is always an integer number. For grayscale image compression, 8 bits/pixel, the **MT** requires a variable of more than 8 bits. Compilers allow declaring variables of type *short* of 16 bit integer signed numbers.

Hence, the total number of bytes required by the **MT** algorithm is

$$2\left(3d^2 + (h\_i)(w\_i)\right). \tag{32}$$

One can observe that this value depends on the image size and on the size of the image subblock chosen for the image transformation process.

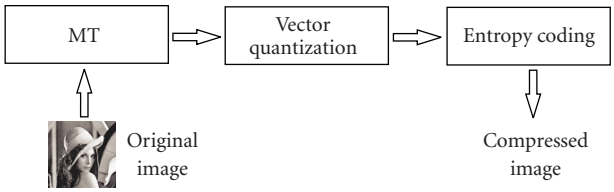

Figure 5: Lossy image compression scheme.

4. EXPERIMENTAL RESULTS

In this section, we present the experimental results obtained using **MT** in an image compression system. First, we compare the **MT** performance when a vector quantization with different sizes from codebook is used. Second, we compare the performance of **MT** when various coding algorithms are used. Finally, we compare the performance to traditional methods of transformation, DCT and DWT. For this purpose, a set of five test grayscale 512×512 pixel images represented by 8 bits/pixel: Lena, Peppers, Elaine, Boat, and Goldhill, was used in simulations.

In our experiments, a lossy image compression scheme has been used, see Figure 5.

In order to measure the **MT** performance, we used a popular objective performance criterion called the peak signal-to-noise ratio (PSNR), which is defined as

$$\text{PSNR} = 10\log_{10}\left(\frac{(2^n-1)^2}{1/M\sum_{i=1}^{M}(p_i-\widetilde{p}_i)^2}\right), \qquad (33)$$

where n is the number of bits per pixel, M is the number of pixels in the image, p_i is the ith pixel in the original image, and $\widetilde{p}_i$ is the ith pixel in the reconstructed image.

The first experiment includes only first two stages of the system shown in Figure 5: the **MT** and the vector quantization (VQ). The VQ causes the loss of information in the image. This experiment has the objective of analyzing how the **IMT** process reduces data degradation caused by the quantization process. The quantization stage uses the VQ by Linde-Buzo-Gray (LBG) multistage algorithm [36]. The LBG algorithm determines the first codebook, and then each image vector of the image data is encoded by the code vector within the first codebook that best approximates the vector within the image data.

Table 1 and Figure 6 show the obtained PSNR values of test images when the vector quantization of **MT** images with various codebook sizes was used. Figure 7 shows the visual results of this process on Lena, Peppers and Boat images.

In the second experiment, the performance of diverse standard encoding methods applied to the image transformed with **MT** and VQ was evaluated. These methods included statistical modeling techniques, such as arithmetical, Huffman, range, Burrows Wheeler transformation, PPM, dictionary techniques, LZ77 and LZP. The purpose of the second experiment is to analyze **MT** performance in image compression. To this end, a coder that implements LBG VQ and diverse entropy encoding techniques was developed. The

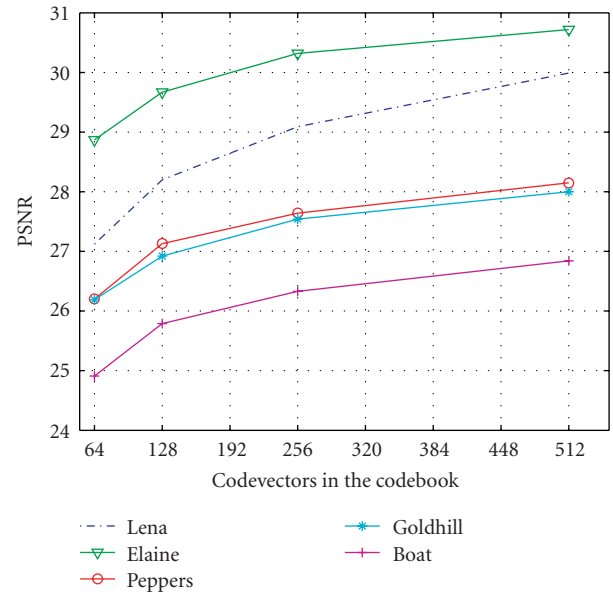

Figure 6: Performance of **MT** on test images with vector quantization with diverse sizes of codebook.

compression performance of our coder on test images is expressed in Table 2.

These results show that the entropy encoding technique, which offers the best results in compression and signal-to-noise ratio are obtained on the image transformed by **MT**, is the PPM coding. The PPM is an adaptive statistical method; its operation is based on partial equalization of chains, that is, the PPM coding predicts the value of an element basing on the sequence of previous elements.

To analyze performance of the image compressor based on **MT**, LBG VQ and PPM coding, we plot in Figure 8 the curves of PSNR versus bit rate (bpp) and PSNR versus compression ratio obtained for test images. In these experiments, the VQ codebook size was varied to achieve different bit rates. One can observe that best performance was achieved for the "Elaine" image.

The transformation stage of an image compressor alone does not produce any information reduction. Its main purpose is to facilitate the information compression at the next stages. Tables 1, 2, and 3 allow comparing the results obtained with the proposed compression scheme formed by **MT**, LBG VQ, and PPM coding, and the same scheme omitting the transformation stage, **MT**. As it was expected, the use of the **MT** considerably improves the compression ratio and in some cases improves the signal-to-noise ratio.

In the third experiment, the efficiency of the image coder based on **MT**, LBG VQ, and PPM coding was compared to that of other image compression methods, JPEG [37, 38], DCT-based embedded coder [37], EZW [11, 38], SPIHT [12, 38], EBCOT [13]. The obtained results show that the proposed method is competitive with the known techniques in the compression ratio and the signal-to-noise ratio. Table 4 presents the comparative results of our coder (**MT**, LBG VQ and PPM) and traditional image compression

Table 1: Performance of **MT** on test images with vector quantization with diverse sizes of codebook.

Image	Performance of **MT** (PSNR) VQ LBG multistage			
	64 codevectors	128 codevectors	256 codevectors	512 codevectors
Lena	27.12	28.20	29.09	29.99
Elaine	28.87	29.67	30.32	30.72
Peppers	26.20	27.13	27.64	28.15
Goldhill	26.19	26.92	27.54	28.00
Boat	24.91	25.79	26.33	26.84

Figure 7: **MT** with VQ on test images: column (a) 64 codevectors, column (b) 128 codevectors, column (c) 256 codevectors, column (d) 512 codevectors.

methods applied to the test image Lena. Figure 9 shows these results as PSNR versus bit rate plots.

Finally, we analyze the number and type of operations and the amount of memory used by the **MT** and the traditional transformation methods. First, we analyze the efficient DCT implementation proposed by Arai et al. [3]. The number of operations used by this algorithm to transform an image is

$$\frac{h\_i}{d}\left(\frac{w\_i}{d}\left(d(\text{op})+d(\text{op})\right)\right)\cdot\frac{2(h\_i\times w\_i)}{d}(\text{op}) \tag{34}$$

for $d\times d$ block, where h_i is the image height, w_i is the image width, and op = 29 sums y 5 multiplications.

The space complexity analysis of the DCT algorithm indicates the memory requirements for this algorithm. In order to process image divided by $d \times d$ blocks, the DCT needs one matrix $a[d][d]$, two vectors $b[d]$, $c[d]$, one vector $e[d/2]$, and one matrix DCT$[h_i][w_i]$. Hence, the total number of units required by this algorithm is

$$\begin{aligned}&\text{un\_}a+\text{un\_}b+\text{un\_}c+\text{un\_}e+\text{un\_DCT}\\&\quad=(d)(d)+d+d+d/2+(h\_i)(w\_i)\\&\quad=d^2+\frac{5d}{2}+(h\_i)(w\_i).\end{aligned} \tag{35}$$

The DCT uses floating point operations. Then, the total number of bytes required by the DCT is

$$4\left(d^2+\frac{5d}{2}+(h\_i)(w\_i)\right). \tag{36}$$

Now, we analyze the DWT when it uses Haar filters, the simplest wavelet filters. The total number of operations used

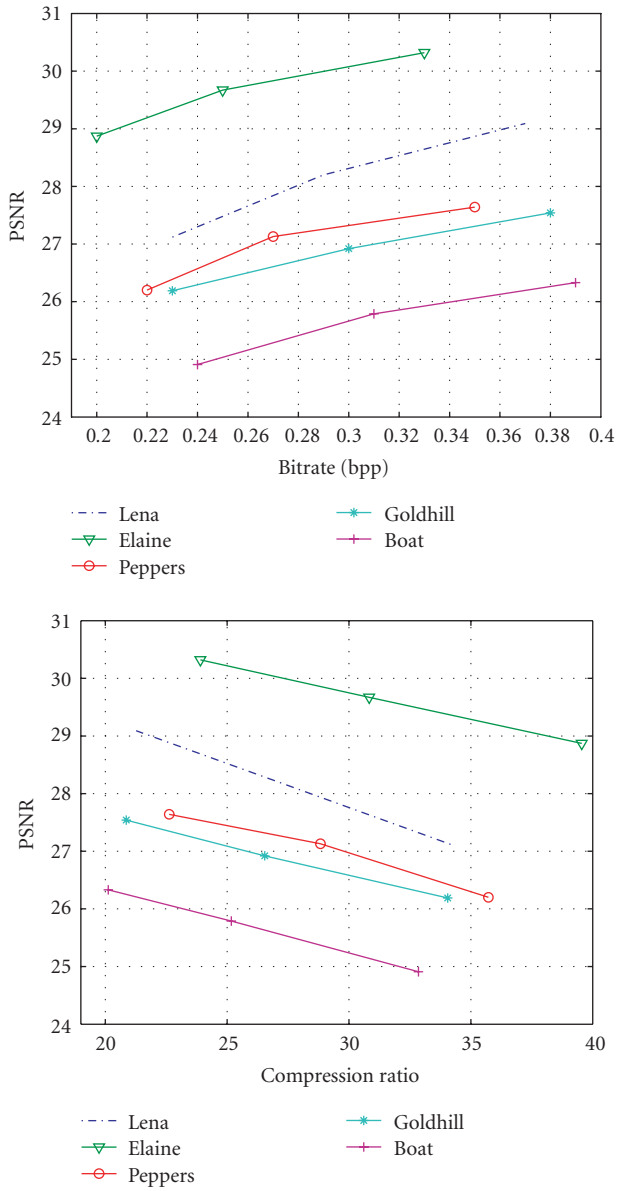

Figure 8: PSNR versus bit rate plots and PSNR versus compression ratio plots for test images when image compressor based on **MT**, LBG VQ, and PPM coding is used.

by the DWT algorithm in this case for image transformation is

$$N1 \text{ operations} + N2 \text{ operations} + \cdots + Nn \text{ operations}. \tag{37}$$

For the DWT of 3 scales:

$$\frac{h\_i}{2}\left(\frac{w\_i}{2}(\text{op})\right) + \frac{h\_i}{4}\left(\frac{w\_i}{4}(\text{op})\right) + \frac{h\_i}{8}\left(\frac{w\_i}{8}(\text{op})\right) \times \frac{\dim}{4}(\text{op}) + \frac{\dim}{16}(\text{op}) + \frac{\dim}{64}(\text{op}). \tag{38}$$

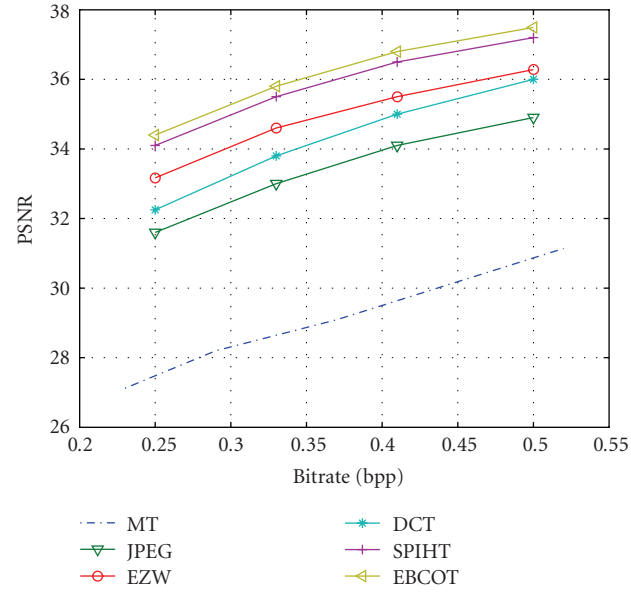

Figure 9: Performance of **MT** (with LBG VQ and PPM coding) and traditional methods on test image Lena.

Generalizing for n scales

$$(\text{op})\sum_{u=1}^{n}\frac{\dim}{2^{u+u}}, \tag{39}$$

where dim $= w\_i \times h\_i$ and op $= 12$ sums y 8 multiplications.

The space complexity analysis of the DWT algorithm specifies memory requirements for this algorithm operation. In order to process an image, the DWT needs two matrices $a[h\_i][w\_i]$ and DWT$[h_i][w_i]$. Then, the total number of bytes required by the DWT is

$$\begin{aligned}\text{un\_}a + \text{un\_DWT} &= (h\_i)(w\_i) + (h\_i)(w\_i)\\ &= 2(h\_i)(w\_i).\end{aligned} \tag{40}$$

The DWT normally uses floating point operations. Then, the total number of bytes required by the DWT is

$$8(h\_i)(w\_i). \tag{41}$$

Finally, the operations number that the **MT** needs in order to transform an image depends on the image size and the image subblock size:

$$\frac{h\_i}{d}\left(\frac{w\_i}{d}(k(d(d(\text{op}))))\right)\cdot(h\_i \times w\_i)(k)(\text{op}), \tag{42}$$

where d is the image vector size, k is the number of elements of the fundamental set of associations, and op $= 1$ sum and 1 comparison.

The space complexity analysis of the **MT** algorithm is expressed by (32).

Based on the previous analysis, we can generate a comparative table of the operations and memory required by

Table 2: Performance comparison of several entropy encoding technique on the information obtained from **MT** on test images.

Image	Entropy encoding technique		Codevectors (VQ LBG multistage)		
			64	128	256
Lena	PPM	Compression ratio	34.17 : 1	27.11 : 1	21.26 : 1
		Bit rate	0.23 bpp	0.29 bpp	0.37 bpp
	Burrows W.	Compression ratio	33.11 : 1	26.33 : 1	20.56 : 1
		Bit rate	0.24 bpp	0.30 bpp	0.38 bpp
	LZP	Compression ratio	30.98 : 1	25.56 : 1	20.35 : 1
		Bit rate	0.25 bpp	0.31	0.39 bpp
	LZ77	Compression ratio	30.23 : 1	24.21 : 1	19.21 : 1
		Bit rate	0.26 bpp	0.33 bpp	0.41 bpp
	Range	Compression ratio	23.86 : 1	20.24 : 1	17.27 : 1
		Bit rate	0.33 bpp	0.39 bpp	0.46 bpp
Boat	PPM	Compression ratio	32.85 : 1	25.17 : 1	20.12 : 1
		Bit rate	0.24 bpp	0.31 bpp	0.39 bpp
	Burrows W.	Compression ratio	32.40 : 1	25.05 : 1	19.88 : 1
		Bit rate	0.24 bpp	0.32 bpp	0.40 bpp
	LZP	Compression ratio	30.56 : 1	23.95 : 1	19.32 : 1
		Bit rate	0.26 bpp	0.33 bpp	0.41 bpp
	LZ77	Compression ratio	30.06 : 1	23.81 : 1	19.46 : 1
		Bit rate	0.26 bpp	0.33 bpp	0.41 bpp
	Range	Compression ratio	27.15 : 1	22.19 : 1	18.72 : 1
		Bit rate	0.29 bpp	0.36 bpp	0.42 bpp
Elaine	PPM	Compression ratio	39.55 : 1	30.83 : 1	23.90 : 1
		Bit rate	0.20 bpp	0.25 bpp	0.33 bpp
	Burrows W.	Compression ratio	36.70 : 1	28.84 : 1	22.74 : 1
		Bit rate	0.21 bpp	0.27 bpp	0.35 bpp
	LZP	Compression ratio	34.90 : 1	28.04 : 1	22.32 : 1
		Bit rate	0.22 bpp	0.28 bpp	0.35 bpp
	LZ77	Compression ratio	33.00 : 1	26.49 : 1	21.04 : 1
		Bit rate	0.24 bpp	0.30 bpp	0.38 bpp
	Range	Compression ratio	27.65 : 1	23.16 : 1	19.45 : 1
		Bit rate	0.28 bpp	0.34 bpp	0.41 bpp
Goldhill	PPM	Compression ratio	34.05 : 1	26.54 : 1	20.86 : 1
		Bit rate	0.23 bpp	0.30 bpp	0.38 bpp
	Burrows W.	Compression ratio	33.22 : 1	26.19 : 1	20.62 : 1
		Bit rate	0.24 bpp	0.30 bpp	0.38 bpp
	LZP	Compression ratio	31.31 : 1	25.08 : 1	20.02 : 1
		Bit rate	0.25 bpp	0.31 bpp	0.39 bpp
	LZ77	Compression ratio	30.87 : 1	24.88 : 1	20.12 : 1
		Bit rate	0.25 bpp	0.32 bpp	0.39 bpp
	Range	Compression ratio	26.42 : 1	22.48 : 1	19.11 : 1
		Bit rate	0.30 bpp	0.35 bpp	0.41 bpp
Peppers	PPM	Compression ratio	35.72 : 1	28.83 : 1	22.62 : 1
		Bit rate	0.22 bpp	0.27 bpp	0.35 bpp
	Burrows W.	Compression ratio	34.33 : 1	27.76 : 1	21.79 : 1
		Bit rate	0.23 bpp	0.28 bpp	0.36 bpp
	LZP	Compression ratio	32.12 : 1	26.63 : 1	21.39 : 1
		Bit rate	0.24 bpp	0.30 bpp	0.37 bpp
	LZ77	Compression ratio	31.19 : 1	25.70 : 1	20.35 : 1
		Bit rate	0.25 bpp	0.31 bpp	0.39 bpp
	Range	Compression ratio	25.20 : 1	21.78 : 1	18.53 : 1
		Bit rate	0.31 bpp	0.36 bpp	0.43 bpp

Table 3: Compression ratio and PSNR obtained by the image compression scheme without **MT**.

Image	Entropy encoding technique		Codevectors (VQ LBG multistage) 64 (PSNR 26.66)	128 (PSNR 27.28)	256 (PSNR 27.57)
Lena	PPM	Compression ratio	14.23 : 1	11.50 : 1	9.43 : 1
		Bit rate	0.56 bpp	0.69 bpp	0.84 bpp
	Burrows W.	Compression ratio	13.79 : 1	11.27 : 1	8.94 : 1
		Bit rate	0.58 bpp	0.71 bpp	0.89 bpp
	Range	Compression ratio	11.98 : 1	10.08 : 1	8.61 : 1
		Bit rate	0.66 bpp	0.79 bpp	0.92 bpp
			Codevectors (VQ LBG multistage) 64 (PSNR 28.95)	128 (PSNR 29.61)	256 (PSNR 30.10)
Goldhill	PPM	Compression ratio	14.23 : 1	11.40 : 1	9.35 : 1
		Bit rate	0.56 bpp	0.70 bpp	0.85 bpp
	Burrows W.	Compression ratio	13.76 : 1	11.29 : 1	8.96 : 1
		Bit rate	0.58 bpp	0.70 bpp	0.89 bpp
	Range	Compression ratio	12.34 : 1	10.46 : 1	8.90 : 1
		Bit rate	0.64 bpp	0.76 bpp	0.89 bpp
			Codevectors (VQ LBG multistage) 64 (PSNR 25.81)	128 (PSNR 27.64)	256 (PSNR 27.75)
Peppers	PPM	Compression ratio	15.06 : 1	12.12 : 1	9.82 : 1
		Bit rate	0.53 bpp	0.65 bpp	0.81 bpp
	Burrows W.	Compression ratio	14.52 : 1	11.83 : 1	9.62 : 1
		Bit rate	0.55 bpp	0.67 bpp	0.83 bpp
	Range	Compression ratio	12.43 : 1	10.56 : 1	8.97 : 1
		Bit rate	0.64 bpp	0.75 bpp	0.89 bpp

Table 4: Comparison between image compression based on **MT**, VQ LBG, and a PPM coding and traditional methods on test image Lena.

Test image Lena: 512 × 512 pixels, 8 bits/pixel	Bit rate	PSNR
Baseline JPEG [37, 38]	0.25	31.6
	0.50	34.9
DCT-based embedded coder [37]	0.25	32.25
	0.50	36.0
EZW [11, 38]	0.25	33.17
	0.50	36.28
SPIHT [12, 38]	0.25	34.1
	0.50	37.2
EBCOT [13]	0.25	34.40
	0.50	37.49
Morphological transform (**MT**)	0.23	27.12
	0.52	31.14

the **MT**, DCT y WDT in order to transform a grayscale image of size 512 × 512 pixels, 8 bits/pixel (see Table 5).

In order to draw conclusions based on these results, it is necessary to consider the following aspects:

(i) at least an operator of the multiplications must be of a real number,

(ii) the Haar filters are the simplest wavelet filters. Normally, the standard schemes of image compression use wavelet filters of greater complexity; a significant example is the JPEG 2000, the filters bank of this standard is formed by a 9-tap low-pass FIR filter and a 7-tap high-pass FIR filter [39] derived from the Daubechies wavelet [40],

(iii) operations used by **MT** are simpler than those required by DCT and DWT,

(iv) all variables used in operations for the **MT** calculation are of the integer type.

(v) The memory required during the **MT** calculation is smaller to the memory needed by DCT or DWT.

On the ground of these considerations and the obtained results, with respect to the processing speed and the memory requirements, it can be concluded that the **MT** proves to be a more efficient algorithm than the traditional methods.

Table 5: Operations and memory required by **MT**, DCT y DWT in order to transform a grayscale image of 512 × 512 pixels: 8 bits/pixel.

Transform	Input data type	Output data type	Required memory (bytes)	Number and type of operations
DCT Blocks of 8 × 8	Integer	Float	1,048,912	1,900,544 sums, 327,680 multiplications
DWT Haar filters, 3 scales	Integer	Float	2,097,152	1,032,192 sums, 688,128 multiplications
MT Blocks of 8 × 8	Integer	Integer	524,672	2,097,152 sums, 2,097,152 comparisons

5. CONCLUSIONS

The use of morphological associative memories at the transformation stage of an image compressor has demonstrated a high competitiveness in its efficiency in comparison to traditional methods based on DCT (JPEG) or DWT (EZW, SPIHT, and EBCOT). Moreover, **MT** has low-computational complexity since its operation is based on maximums or minimums of sums, that is, MAM uses only operations of sums and comparisons. This fact results in the high-processing speed and low demand of resources (system memory). Indeed, to calculate a morphological associative memory for an image block of 8 × 8 pixels, 512 sums and 512 comparisons are required.

The quantification process introduces random noise in the transformed image, so the **MT** uses the HMM *min* or HMM *max*. And the MAMs do not perform well when the patterns contain erosive and dilative noise at the same time, thus limiting the **MT** ability to attenuate the noise induced by the quantifier. To resolve this problem and obtain a better response in the signal-to-noise ratio, it is possible to use alternative schemes of associative memories robust to random noise in the input patterns. This aspect is the subject of future work.

ACKNOWLEDGMENT

This work was partially supported by Instituto Politecnico Nacional as a part of the research project SIP no. 20080903.

REFERENCES

[1] N. Ahmed, T. Natrajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computer*, vol. 23, no. 1, pp. 90–93, 1974.

[2] W.-H. Chen, C. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. 25, no. 9, pp. 1004–1009, 1977.

[3] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for image," *Transactions of the IEICE*, vol. E-71, no. 11, pp. 1095–1097, 1988.

[4] B. D. Tseng and W. C. Miller, "On computing the discrete cosine transform," *IEEE Transactions on Computers*, vol. 27, no. 10, pp. 966–968, 1978.

[5] S. Winograd, "On computing the discrete Fourier transform," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 73, no. 4, pp. 1005–1006, 1976.

[6] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[7] ISO, "Digital compression and coding of continuous-tone still images: requirements and guidelines," 1994, ISO/IEC IS 10918-1.

[8] R. A. DeVore, B. Jawerth, and B. J. Lucier, "Image compression through wavelet transform coding," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 719–746, 1992.

[9] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions of Image Processing*, vol. 1, no. 2, pp. 205–220, 1992.

[10] A. S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform," *IEEE Transactions of Image Processing*, vol. 1, no. 2, pp. 244–250, 1992.

[11] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, 1993.

[12] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.

[13] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, 2000.

[14] Joint Photographic Experts Group, 2000, JPEG 2000 Part I Final Committee Draft Version 1.0 ISO/IEC JTC 1/SC 29/WG 1, N1646R, (ITU-T SG8).

[15] T. Kohonen, "Automatic formation of topological maps of patterns in a self-organizing system," in *Proceedings of the 2nd Scandinavian Conference on Image Analysis (SCIA '81)*, E. Oja and O. Simula, Eds., pp. 214–220, Suomen Hahmontunnistustutkimuksen seura r.y., Helsinki, Finland, June 1981.

[16] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[17] A. Bogdan and H. E. Meadows, "Kohonen neural network for image coding based on iteration transformation theory," in *Neural and Stochastic Methods in Image and Signal Processing*, vol. 1766 of *Proceedings of SPIE*, pp. 425–436, San Diego, Calif, USA, July 1992.

[18] C. Amerijckx, M. Verleysen, P. Thissen, and J.-D. Legat, "Image compression by self-organized Kohonen map," *IEEE Transactions on Neural Networks*, vol. 9, no. 3, pp. 503–507, 1998.

[19] C. Amerijckx, J.-D. Legat, and M. Verleysen, "Image compression using self-organizing maps," *Systems Analysis Modelling Simulation*, vol. 43, no. 11, pp. 1529–1543, 2003.

[20] M. Mokhtari and A. Boukelif, "Optimization of fractal image compression based on Kohonen neural networks," in

Proceedings of the 2nd International Symposium on Control, Communications, and Signal Processing (ISCCSP '06), Marrakech, Morocco, March 2006.

[21] S. Panchanathan, T. H. Yeap, and B. Pilache, "Neural network for image compression," in *Applications of Artificial Neural Networks III*, vol. 1709 of *Proceedings of SPIE*, pp. 376–385, Orlando, Fla, USA, April 1992.

[22] R. Setiono and G. Lu, "Image compression using a feed-forward neural network," in *Proceedings of the IEEE World Congress on Computational Intelligence*, vol. 7, pp. 4761–4765, Orlando, Fla, USA, June-July 1994.

[23] Q. Ji, "Image compression using a self-organized neural network," in *Applications of Artificial Neural Networks in Image Processing II*, vol. 3030 of *Proceedings of SPIE*, pp. 56–59, San Jose, Calif, USA, February 1997.

[24] S. B. Roy, K. Kayal, and J. Sil, "Edge preserving image compression technique using adaptive feed forward neural network," in *Proceedings of the IASTED European International Conference on Internet and Multimedia Systems and Applications (EuroIMSA '05)*, pp. 467–471, Grindelwald, Switzerland, February 2005.

[25] K. S. Ng and L. M. Cheng, "Artificial neural network for discrete cosine transform and image compression," in *Proceedings of the 4th International Conference on Document Analysis and Recognition (ICDAR '97)*, vol. 2, pp. 675–678, Ulm, Germany, August 1997.

[26] C. J. C. Burges, H. S. Malvar, and P. Y. Simard, "Improving wavelet image compression with neural networks," Tech. Rep. MSR-TR-2001-47, Microsoft Research, Redmond, Wash, USA, 2001.

[27] H. Nait-Charif and F. M. Salam, "Neural networks-based image compression system," in *Proceedings of the 43rd IEEE Symposium on Midwest Circuits and Systems (MWSCAS '00)*, vol. 2, pp. 846–849, Lansing, Mich, USA, August 2000.

[28] P. Danchenko, F. Lifshits, I. Orion, S. Koren, A. D. Solomon, and S. Mark, "NNIC—neural network image compressor for satellite positioning system," *Acta Astronautica*, vol. 60, no. 8-9, pp. 622–630, 2007.

[29] G. X. Ritter, D. Li, and J. N. Wilson, "Image algebra and its relationship to neural networks," in *Aerospace Pattern Recognition*, vol. 1098 of *Proceedings of SPIE*, pp. 90–101, Orlando, Fla, USA, March 1989.

[30] G. X. Ritter and P. Sussner, "An introduction to morphological neural networks," in *Proceedings of the 13th International Conference on Pattern Recognition (ICPR '96)*, vol. 4, pp. 709–717, Vienna, Austria, August 1996.

[31] G. X. Ritter, P. Sussner, and J. L. Diaz-de-León, "Morphological associative memories," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 281–293, 1998.

[32] P. Sussner and M. E. Valle, "Gray-scale morphological associative memories," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 559–570, 2006.

[33] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, 1982.

[34] J. Serra, Ed., *Image Analysis and Mathematical Morphology, Volume 2: Theoretical Advances*, Academic Press, Boston, Mass, USA, 1988.

[35] E. Guzmán, O. Pogrebnyak, C. Yánez, and J. A. Moreno, "Image compression algorithm based on morphological associative memories," in *Proceedings of the 11th Iberoamerican Congress in Pattern Recognition (CIARP '06)*, vol. 4225 of *Lecture Notes in Computer Science*, pp. 519–528, Springer, Cancun, Mexico, November 2006.

[36] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[37] Z. Xiong, O. G. Guleryuz, and M. T. Orchard, "A DCT-based embedded image coder," *IEEE Signal Processing Letters*, vol. 3, no. 11, pp. 289–290, 1996.

[38] Z. Xiong, K. Ramchandran, M. T. Orchard, and Y.-Q. Zhang, "A comparative study of DCT—and wavelet-based image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 5, pp. 692–695, 1999.

[39] T. Acharya and P.-S. Tsai, *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures*, John Wiley & Sons, New York, NY, USA, 2005.

[40] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 961–1005, 1990.