World's largest Science, Technology & Medicine Open Access book publisher

3,300+ OPEN ACCESS BOOKS
106,000+ INTERNATIONAL AUTHORS AND EDITORS
113+ MILLION DOWNLOADS
BOOKS DELIVERED TO 151 COUNTRIES
AUTHORS AMONG TOP 1% MOST CITED SCIENTIST
12.2% AUTHORS AND EDITORS FROM TOP 500 UNIVERSITIES

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

WEB OF SCIENCE™

Chapter from the book Search Algorithms and Applications
Downloaded from: http://www.intechopen.com/books/search-algorithms-and-applications

Interested in publishing with InTechOpen?
Contact us at book.department@intechopen.com
1. Introduction

Vector quantization (VQ) has been used as an efficient and popular method in lossy image and speech compression (Gray, 1984; Nasrabadi & King, 1988; Gersho & Gray, 1992). In these areas, VQ is a technique that can produce results very close to the theoretical limits. The most widely used and simplest technique for designing vector quantizers is the LBG algorithm by Y. Linde et al. (1980). It is an iterative descent algorithm which monotonically decreases the distortion function towards a local minimum. Sometimes, it is also referred to as generalized Lloyd algorithm (GLA), since it is a vector generalization of a clustering algorithm due to Lloyd (1982).

New algorithms for VQ based on associative memories or artificial neuronal networks (ANNs) have arisen as an alternative to traditional methods. Within this approach, many VQ algorithms have been proposed. We mention some of them.

The self-organizing map (SOM) ANN, developed by Prof. Teuvo Kohonen in the early 1980s (Kohonen, 1981; Kohonen 1982), has been used with a great deal of success in creating new schemes for VQ. The SOM is a competitive-learning network. C. Amerijckx et al. (1998) 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 (Amerijckx et al., 2003), the same authors proposed an image compression scheme for lossless compression using SOMs and the same principles.

Eyal Yair et al. (1992) provide a convergence analysis for the Kohonen Learning Algorithm (KLA) with respect to VQ optimality criteria and introduce a stochastic relaxation technique which produces the global minimum but is computationally expensive. By incorporating the
principles of the stochastic approach into the KLA, a new deterministic VQ design algorithm, called the soft competition scheme (SCS), is introduced. The new algorithm is an on-line method in which the codeword update is governed by a “soft” competition between the codevector, which are updated simultaneously for each presentation of a training vector. A new VQ scheme based on competitive learning neural network and LBG vector quantization was presented by Basil and Jiang (1999). The algorithm integrates advantages presented in both LBG vector quantization and competitive learning neural networks to achieve an improved performance in comparison with their existing forms when it is applied to image compression.

Chin-Chuan Han et al. (2006; 2007) proposed an hybrid approach for VQ for obtaining the better codebook. It is modified and improved based on the centroid neural network adaptive resonance theory (CNN-ART) and the enhanced LBG (Linde-Buzo-Gray) approaches. Three modules, a neuronal net (NN) based clustering, a mean shift (MS) based refinement, and a principal component analysis (PCA) based seed assignment, are repeatedly utilized. Basically, the seed assignment module generates a new initial codebook to replace the low utilized codewords during the iteration. The NN-based clustering module clusters the training vectors using a competitive learning approach. The clustered results are refined using the mean shift operation.

A fuzzy Learning Vector Quantization (LVQ) which is based on the fuzzification of LVQ was proposed by Yong-Soo and Sung-Ihl (2007). The proposed fuzzy LVQ uses the different learning rate depending on whether classification is correct or not. When the classification is correct, it uses the combination of a function of the distance between the input vector and the prototypes of classes and a function of the number of iteration as the fuzzy learning rate. On the other hand, when the classification is not correct, it uses the combination of the fuzzy membership value and a function of the number of iteration as the fuzzy learning rate. The proposed FLVQ (fuzzy LVQ) is integrated into the supervised IAFC (Integrated Adaptive Fuzzy Clustering) neural network 5. The iris data set was used to compare the performance of the supervised IAFC neural network 5 with those of LVQ algorithm and back propagation neural network.

VQ is a process in which data to be encoded are divided into small blocks, forming vectors, which are then sequentilly encoded vector by vector. The VQ process is divided in two phases: codebook design and encoding phases.

In the codebook design phase the idea is to identify a set, named codebook, of possible vectors (codewords) which are representative of the information to be encoded. The encoding phase searches for codeword with the closest match for every given input vector. The final encoding is then simply a process of sequentially listing the codeword indexes which were deemed to most closely match the vectors making up the original data.

One of the most serious problems for VQ, especially for high dimensional vectors, is the high computational complexity of searching for the closest codeword in the codebook design and encoding phases. Although quantizing high dimensional vectors rather than low dimensional vectors result in a better performance, the computation time needed for vector quantization grows exponentially with the vector dimension. This makes high dimensional vectors unsuitable for vector quantization. Thus, Given a data set (codebook) of \( m \) codewords, \( \{c_1, c_2, \ldots, c_m\} \), the problem is to find the \( i \) element that is closest to a input vector, \( x \), where \( x, c_i \in \mathbb{R}^d \forall i \).
In this study, to overcome the said problem an efficient high-speed search algorithm for image quantization based on Extended Associative Memories (EAM) is proposed. This new algorithm significantly reduces the computation needed without sacrificing performance. A preliminary version of this work first appeared in (Guzmán et al., 2008). The present work is a widespread version of that one. The new results that this work includes are: 1) A mathematical analysis of the influence that each operator of our proposal has in the codebook design and encoding phase performance. 2) The results of the proposed algorithm when additional operators, \textit{pmed} and \textit{sum}, are used. 3) A complexity analysis of the proposed algorithm for additional operators. 4) The results that let us evaluate the influence that the proposed VQ algorithm has in the image compression performance.

The remaining sections of this work are organized as follows. In next Section, a brief theoretical background of extended associative memories is given. In Section 3 we describe our proposal, the high-speed search algorithm for image quantization based on EAM; furthermore, this section include a complexity analysis of the proposed algorithm. Numerical simulation results obtained for the conventional image quantization techniques (LBG algorithm) and the proposed algorithm, when it uses \textit{prom}, \textit{med} and \textit{pmed} operators, are provided and discussed in Section 4. Finally, Section 5 contains the conclusions of this study.

2. Extended associative memories model

An associative memory designed for pattern classification is an element whose fundamental purpose is to establish a relation of an input pattern $\mathbf{x} = [x_i]_n$ with the index $i$ of a class $c_i$ (see Fig. 1).

$$
\mathbf{x} = \begin{pmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{pmatrix}
$$

Fig. 1. Associative memory scheme for pattern classification.

Let $\{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \ldots, (\mathbf{x}_N, c_N)\}$ be $N$ couples of a pattern and its corresponding class index forming the fundamental set of couples composed by patterns and their corresponding class indices:

$$
\left\{ (\mathbf{x}^\mu, c_\mu); \mu = 1, 2, \ldots, N \right\}
$$

where $\mathbf{x}^\mu \in \mathbb{R}^n$ and $c_\mu = 1, 2, \ldots, N$. The associative memory is represented by a matrix generated from the fundamental set of couples and denoted by $\mathbf{M}$.

H. Sossa et al. (2004) proposed an associative memory model for the classification of real-valued patterns. This model, named Extended Associative Memory (EAM), is an extension of the Lernmatrix model proposed by K. Steinbuch (1961). The EAM is based on the general concept of the associative memory learning function and presents a high performance in pattern classification of real value data by its components and with altered patterns (Vázquez, 2005; Barron, 2006).
In the EAM, being an associative memory utilized for pattern classification, all the synaptic weights that belong to output $i$ are accommodated at the $i$-th row of a matrix $M$. The final value of this row is a function $\phi$ of all the patterns belonging to class $i$. The function $\phi$ acts as a generalizing learning mechanism that reflects the flexibility of the memory (Sossa et al., 2004).

### 2.1 Training stage of the EAM

The training stage of the EAM consists of evaluating function $\phi$ for each class. Then, the matrix $M$ can be structured as:

$$M = \begin{bmatrix} \varphi_1 & \cdots & \varphi_N \end{bmatrix}$$  \hspace{1cm} (2)

where $\varphi_i$ is the evaluation of $\phi$ for all patterns of $i$-th class, $i = 1, 2, \ldots, N$. The function $\phi$ can be evaluated in various manners. The arithmetical average operator ($\text{prom}$) is frequently used in the signals and images treatment. In (Sossa et al., 2004), the authors studied the performance of this operator and the median operator ($\text{med}$) to evaluate the function $\phi$. Furthermore, in (Vázquez, 2005) the use of the average point operator ($\text{pmed}$) and sum operator ($\text{sum}$) were proposed to evaluate the function $\phi$.

The goal of the training stage is to establish a relation between an input pattern $x = [x_i]_n$, and the index $i$ of a class $c_i$.

Considering that each class is composed for $q$ patterns $x = [x_i]_n$, and $\varphi_i = [\varphi_{i,1}, \ldots, \varphi_{i,n}]$. Then, the training stage of the EAM, when the $\text{prom}$ operator is used to evaluate the function $\varphi_i$, is defined as:

$$\phi_{i,j} = \frac{1}{q} \sum_{l=1}^{q} x_{j,l}, \hspace{0.5cm} j = 1, \ldots, n$$  \hspace{1cm} (3)

The training stage of the EAM, when the $\text{med}$ operator is used to evaluate the function $\varphi_i$, is defined as:

$$\phi_{i,j} = \text{med}_{l=1}^{q} x_{j,l}, \hspace{0.5cm} j = 1, \ldots, n$$  \hspace{1cm} (4)

When the $\text{pmed}$ operator is used to evaluate the function $\varphi_i$, the training stage of the EAM is defined as:

$$\phi_{i,j} = \frac{\gamma_{i,j} + \lambda_{i,j}}{2}, \hspace{0.5cm} j = 1, \ldots, n$$  \hspace{1cm} (5)

When the $\text{sum}$ operator is used to evaluate the function $\varphi_i$, the training stage of the EAM is defined as:

$$\phi_{i,j} = \gamma_{i,j} + \lambda_{i,j}$$  \hspace{1cm} (6)

where $\gamma_{ij}$ and $\lambda_{ij}$ represent the maximum and minimum vectors of the class $i$ respectively. These vectors are defined as...
\[ y_{i,j} = \bigvee_{l=1}^{q} (x_{l,j}^{i}) \]  
\[ \lambda_{i,j} = \bigwedge_{l=1}^{q} (x_{l,j}^{i}) \]  
\[ \gamma_{i,j} = \bigvee_{l=1}^{q} (x_{l,j}^{i}) \]  
\[ \lambda_{i,j} = \bigwedge_{l=1}^{q} (x_{l,j}^{i}) \]

The operators \( \max (\bigvee) \) and \( \min (\bigwedge) \) perform the morphological operations on the patterns belongs to each class.

The associative memory, \( \mathbf{M} \), is obtained after evaluating all functions \( \phi_{j} \). In the case when \( N \) classes exist and the vectors to classify are \( n \)-dimensional, the resultant memory \( \mathbf{M} = [m_{ij}]_{N \times n} \) is denoted as:

\[
\mathbf{M} = \begin{bmatrix}
\phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,n} \\
\phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
\phi_{N,1} & \phi_{N,2} & \cdots & \phi_{N,n}
\end{bmatrix}
= \begin{bmatrix}
m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\
m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
m_{N,1} & m_{N,2} & \cdots & m_{N,n}
\end{bmatrix}
\]  
\[ (9) \]

### 2.2 Classification stage of the EAM

The goal of the classification stage is the generation of the class index to which an input pattern belongs.

The pattern classification by EAM is done when a pattern \( \mathbf{x}^{\mu} \in \mathbb{R}^{n} \) is presented to the memory \( \mathbf{M} \) generated at the training stage. The EAM possess the feature of classifying a pattern not necessarily one of those already used to build the memory \( \mathbf{M} \).

When the \texttt{prom} or \texttt{pmed} operators are used, the class to which \( \mathbf{x} \) belongs is given by

\[
i = \arg \left[ \bigwedge_{i=1}^{N} \bigvee_{j=1}^{n} [m_{ij} - x_{j}] \right] \]  
\[ (10) \]

In this case, the operators \( \vee \equiv \max \) and \( \wedge \equiv \min \) perform morphological operations on the difference of the absolute values of the elements \( m_{ij} \) of \( \mathbf{M} \) and the component \( x_{j} \) of the pattern \( \mathbf{x} \) to be classified.

When the EAM uses the \texttt{med} operator, the class to which \( \mathbf{x} \) belongs is given by

\[
i = \arg \left[ \bigwedge_{i=1}^{N} \bigvee_{j=1}^{n} [m_{ij} - \text{med}_{j=1}^{n} x_{j}] \right] \]  
\[ (11) \]

In this case, the operator \( \bigwedge \equiv \min \) performs a morphological erosion over the absolute difference of the median of the elements \( m_{ij} \) of \( \mathbf{M} \) and the median of the component \( x_{j} \) of the pattern \( \mathbf{x} \) to be classified.

When the \texttt{sum} operator is used, the recovery matrix, \( \mathbf{r} = [r_{ij}]_{N \times n} \), must be generated. The components of the recovery matrix can be computed of two forms:

\[ r_{ij} = x_{j} + \gamma_{i,j} \]  
\[ (12) \]
Then, the class to which \( x \) belongs is given by

\[
i = \arg\left[ \bigwedge_{i=1}^{N} \bigvee_{j=1}^{u} \left| m_{ij} - r_{ij} \right| \right]
\]

In this case, the operators \( \lor \equiv \max \) and \( \land \equiv \min \) perform morphological operations on the difference of the absolute values of the elements \( m_{ij} \) of \( M \) and the component \( r_{ij} \) of the recovery matrix.

The Theorem 1 and Corollaries 1-5 from (Sossa et al., 2004) govern the conditions that must be satisfied to obtain a perfect pattern classification; either the pattern may be from the fundamental set of couples or be an altered version of the pattern. Here we reproduce them.

Theorem 1. Let \( d_i = \lor_{\text{class}j} d(x, \varphi_j) \) and \( R_i = \{x : d(x, \varphi_i) \leq d_i \} \) be hyper cubes centred at \( \varphi_i \) and semi-sides \( d_i \), \( i = 1, \ldots, N \). If \( d(\varphi_i, \varphi_j) > 2\max\{d_i, d_j\} \) then:

i. \( R_i \cap R_j = \emptyset, \ 1 \leq i, j \leq N, \ i \neq j \).

ii. \( x \in R_i \) implies \( d(x, \varphi_i) \leq d(x, \varphi_j) \).

iii. \( x \in R_j \) implies \( d(x, \varphi_j) \leq d(x, \varphi_i) \).

Corollary 1. If the conditions of Theorem 1 hold for all \( 1 \leq i, j \leq m, \ i \neq j \) then:

i. \( R_i \cap R_j = \emptyset, \ 1 \leq i, j \leq N, \ i \neq j \).

ii. If \( x \in R_i \) then \( d(x, \varphi_i) \leq d(x, \varphi_j) \), \( 1 \leq i, j \leq N, \ i \neq j \).

Corollary 2. Perfect classification of the fundamental set of patterns is immediate due to any fundamental pattern \( x \), by constructions, belongs to its corresponding region \( R_i \) and if the conditions of Theorem 1 hold, then \( d(x, \varphi_j) \leq d(x, \varphi_i) \), \( 1 \leq j \leq N, \ i \neq j \). Thus the memory assigns the fundamental pattern \( x \) to class \( i \) as is due.

Corollary 3. If \( \tilde{x} \) is an altered version of fundamental pattern \( x \in R_i \), \( \tilde{x} \) is correctly classified if \( \tilde{x} \in R_i \).

Corollary 4. The attraction basin of \( i \)-th class is at least \( R_i \) and the convergence of correct classification happens in one step.

Corollary 5. Let \( x \) be a pattern to be classified. If \( x \in R_i \) and \( x \not\in R_j \) then it must be classified at the class for which:

i. \( i = \arg\left[ \bigwedge_{i=1}^{N} \bigvee_{j=1}^{u} \left| m_{ij} - x \right| \right] \) prom and pmed operators.

ii. \( i = \arg\left[ \bigwedge_{i=1}^{N} \med_{i=1}^{n} m_{ij} - \med_{j=1}^{n} x \right] \) med operator.

iii. \( i = \arg\left[ \bigwedge_{i=1}^{N} \bigvee_{j=1}^{u} \left| m_{ij} - r_{ij} \right| \right] \) sum operator.
3. Image quantization based on extended associative memories

In this section the image quantization algorithm based on EAM is described. Our proposal uses the EAM in both codebook generation and in the encoding phase. In codebook generation, applying the learning stage of the EAM on a codebook generated by the LBG algorithm a new codebook, named EAM-codebook, is obtained. In encoding phase, we propose a High-Speed Search Algorithm Applied to Image Quantization based on EAM; the VQ process is performed by means of the recalling stage of EAM using as associative memory the EAM-codebook. This process generates a set of the class indices to which each input vector belongs.

3.1 Basic notations and preliminary definitions

Vector quantization can be viewed as a mapping $Q$, from a $n$-dimensional vector space $R^n$ into a finite subset $C$ of $R^n$:

$$Q : R^n \rightarrow C$$

(15)

where $C = \{y^i : i = 1, 2, ..., N\}$ is the set of reconstruction vectors and $N$ is the number of vectors in $C$. Thus,

i. $\bigcup_{i=1}^{N} y^i = C,$

ii. $y^i \cap y^j = \emptyset, \ \forall i, j, i \neq j.$

Each $y^i$, in $C$ is called a codeword and $C$ is called the codebook for the vector quantizer. For every source vector $x$, a codeword $y^i$, in $C$ is selected as the representation for $x$. This process is called the quantization phase (or the codebook search phase) of the vector quantizer, denoted by $Q(x) = y^i$. Then, the codeword $y^i$, is represented by some symbols (normally the address of the codeword in the codebook) and transmitted through the channel. This process is called the encoding phase of the vector quantizer.

On the other side of the channel, the received symbols are used to select the codewords from the codebook to reproduce the source signals. This process is called the decoding phase of the vector quantizer. The average number of bits required to represent the symbols in the encoding phase is the rate of the quantizer, and the average quantization error between input source signals and their reproduction codewords is the distortion of the vector quantizer. Increasing the number of codewords in the codebook can decrease the distortion of a vector quantizer and, normally, will increase the rate also. One major concern for vector quantizer design is the trade-off between distortion and rate.

On the other hand, both codebook generation and encoding phases of the proposed VQ algorithm make use of individual blocks of the image, which must be converted to vectors. Let the image be represented by a matrix, $A = [a_{ij}]_{hi \times wi}$, where $hi$ is the image height and $wi$ is the image width; and $a$ represents the $ij$-th pixel value: $a \in \{0,1,2,...,2^L - 1\}$, where $L$ is the number of bits necessary to represent the value of a pixel.

Now, we define the image block and image vector terms.

**Definition 1**, image block (ib). Let $A = [a_{ij}]$ be a $hi \times wi$ matrix representing an image, and let $ib = [ib_{ij}]$ be a $d \times d$ matrix. The $ib$ matrix is defined as a image block of the $A$ matrix if the $ib$ matrix is a subgroup of the $A$ matrix such that
The value of the pixel determined by the coordinates \((\delta + i, \tau + j)\), where \((\delta, \tau)\) and \((\delta + d, \tau + d)\) are the beginning and the end of the image block, respectively.

**Definition 2.** Image vector \(\text{iv}\). Let \(ib = [ib_{ij}]\) be an image sub-block and let \(iv = [iv_i]\) be a vector of size \(d\). The \(i\)-th row of the \(ib\) matrix is said to be an image vector \(iv\) such that

\[
iv_i = [ib_{i1}, ib_{i2}, \ldots, ib_{id}]
\]

where \(i = 1, 2, 3, \ldots, d\). From each image block, \(d\) image vectors can be obtained:

\[
iv^\mu = [ib_{\mu1}, ib_{\mu2}, \ldots, ib_{\mu d}]
\]

where \(\mu = 1, 2, 3, \ldots, d\).

A image block can be represented as a vector \(x = [x_{ij}]\); thus, \(ib\) can be defined as a set of \(d\) row image vectors \(iv = [iv_1, iv^2, \ldots, iv^d]\), where, \(iv_j = ib_{ij}, j = 1, 2, \ldots, d\); then

\[
x = [iv^i : i = 1, 2, \ldots, d]
\]

where, \(x_p = iv^j, i, j = 1, 2, \ldots, d, \quad p = 1, 2, \ldots, n, \quad n = d \times d\).

Finally, the new image representation is defined as

\[
A = X = \{x^i : i = 1, 2, \ldots, M\}
\]

where, \(M = (hi / d)(wi / d)\).

In both EAM-codebook generation and encoding phases the new image representation is used.

### 3.2 Codebook generation

In this phase, an associative network is generated applying the learning stage of the EAM between a codebook generated by the LBG algorithm and a training set. This associative network is named EAM-codebook and establishes a relation between training set and the LBG codebook.

The codebook generation based on the LBG algorithm and the EAM is fundamental in obtaining a fast search algorithm for image VQ.

The EAM-codebook is computed in three steps:

**Step 1.** **LBG codebook generation.** This step applied the LBG algorithm on the image blocks (training set), \(X = \{x^i : i = 1, 2, \ldots, M\}\) to generate an initial codebook: \(C = y^i : i = 1, 2, \ldots, N\).

This codebook is formed by a set of \(n\)-dimensional vectors \(y = [y_j]\), named *codeword*.

The Fig 2 shows the scheme of the LBG algorithm, where, \(i = 1, \ldots, n, \quad \mu = 1, \ldots, N, \quad s_1, s_2, \ldots, s_N\) can have different value and \(s_1 + s_2 + \ldots + s_N = M\).
Fig. 2. LBG codebook generation.

Step 2. Definition of the fundamental set of couples. This step uses the codebook generated in the previous step and the training set. This step designs a $Q$ mapping and assigns an index $i$ to each $n$-dimensional input pattern $\mathbf{x} = (x_1, x_2, ..., x_n)$, with $Q(\mathbf{x}) = \mathbf{y}^i = (y_{i1}, y_{i2}, ..., y_{im})$. The $Q$ mapping is designed to map $\mathbf{x}$ to $\mathbf{y}^i$ with $\mathbf{y}^i$ satisfying the following condition:

$$d(\mathbf{x}, \mathbf{y}^i) = \min_j d(\mathbf{x}, \mathbf{y}^j), \text{for } j = 1, 2, 3, ..., N$$

(21)

where $d(\mathbf{x}, \mathbf{y}^i)$ is the distortion of representing the input pattern $\mathbf{x}$ by the codeword $\mathbf{y}^i$, measured by Euclidean distance.

Each codeword $\mathbf{y}^i$ represents a class (existing $N$ classes), then a set of $M$ indices that indicates to which class, $\mathbf{y}^i$, each input pattern $\mathbf{x}$ belongs is generated. Let us to denote this set of indices as $H = \{h_i : i = 1, 2, ..., M\}$. This step also generates a set of $N$ indices that indicates the number of input patterns that integrate to each class. This set is denoted as $B = \{b_i : i = 1, 2, ..., N\}$.

Based on the sets $H$ and $B$ and considering that the set of input patterns is integrated from $M$ image blocks, $X = \{\mathbf{x}^i : i = 1, 2, ..., M\}$, the fundamental set of couples is formed

$$\begin{align*}
\mathbf{x}^{h_{11}} &\in C_1 & \mathbf{x}^{h_{12}} &\in C_2 & \cdots & \mathbf{x}^{h_{1M}} &\in C_N \\
\mathbf{x}^{h_{21}} &\in C_1 & \mathbf{x}^{h_{22}} &\in C_2 & \cdots & \mathbf{x}^{h_{2M}} &\in C_N \\
\vdots & & \vdots & & \ddots & \vdots & \vdots \\
\mathbf{x}^{h_{b_1}} &\in C_1 & \mathbf{x}^{h_{b_2}} &\in C_2 & \cdots & \mathbf{x}^{h_{b_N}} &\in C_N
\end{align*}$$

(22)

where, $b_1, b_2, ..., b_N$ can have different value and $b_1 + b_2 + ... + b_N = M$. 

www.intechopen.com
The Fig. 3 shows the definition of the fundamental set of couples.

**Step 3.** *Generation of a new codebook based on EAM.* The goal of the third step is to generate the EAM-codebook.

Finding an optimal solution of the feature vector quantization problem necessitates a training process which involves learning the probability distribution of the input data. For this purpose, the training stage of the EAM is applied on the fundamental set of couples, denoted by the equation (22).

Considering that each class can group \( b_1,b_2,\ldots,b_N \) input vectors, \( \mathbf{x} = [x_i]_n \), the generalizing learning mechanism of the EAM is defined by the expressions (3), (4), (5) and (6) for the operators **prom**, **med**, **pmed** and **sum** respectively.

\[
\begin{align*}
    m_{av} &= \frac{1}{q} \sum_{g=1}^{q} x_v^g \\
    m_{av} &= \text{med} x_v^g \\
    m_{av} &= \left( \bigvee_{g=1}^{q} x_v^{g*} + \bigwedge_{g=1}^{q} x_v^{g*} \right) / 2 \\
    m_{av} &= \left( \bigvee_{g=1}^{q} x_v^{g*} + \bigwedge_{g=1}^{q} x_v^{g*} \right)
\end{align*}
\]
where \( q \) can take the value of \( b_1, b_2, \ldots, b_N \), \( v = 1, \ldots, n \), and \( u = 1, 2, \ldots, N \).

The result of this step is an associative network that establishes a relation between the \( q \) input vectors and the class to which they belong.

Since \( N \) classes exist and the input vectors that integrate them are \( n \)-dimensional, then the associative network is represented by a matrix \( \mathbf{M} = [m_{uv}]_{N \times n} \), which is the EAM-codebook:

\[
\text{EAM-codebook} = \mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & \cdots & m_{Nn} \end{bmatrix} = [m_{uv}]_{N \times n}
\] (23)

The column \( \{m_{u1}, m_{u2}, \ldots, m_{un}\} \) of \( \mathbf{M} \) represents the centroid of the class \( u \), \( u = 1, 2, \ldots, N \).

That is, the synaptic weight, \( m_{uv} \), is a function \( \phi \) of all elements belonging to class \( u \). These synaptic weights determine the behaviour of the net.

### 3.3 High-speed search algorithm for image quantization based on EAM

In the encoding phase of VQ process each input pattern is replaced by the codeword index that presents the nearest matching based on the similarity criterion between input patterns and codewords. In our proposal, this criterion has been codified in the associative memory (EAM-codebook).

The EAM-codebook generation is based on the EAM training stage. Therefore, using the EAM classification stage, the index class to which each input vector belongs is obtained; at the end of this step the image VQ process is completed.

In a general case, when \( N \) classes exist, the input vectors, \( \mathbf{x} \), are \( n \)-dimensional and the prom or pmed operator is used to generate the memory \( \mathbf{M} \), the index class to which an input vector belongs is determined using the morphologic operators max and min

\[
\text{index class} = \arg \left[ \bigwedge_{h} \left[ \bigvee_{j=1}^{n} \vert m_{bj} - x_j \vert \right] \right]
\]

The morphologic operation \( \bigvee_{j=1}^{n} \vert m_{bj} - x_j \vert = d(\mathbf{x}, m_h) \) is the metrics that indicates the distance between each row of \( \mathbf{M} \) and the feature vector \( \mathbf{x} \), that is to say, the degree of belonging of the feature vector \( \mathbf{x} \) with each one of the \( N \) classes; that is

\[
d(\mathbf{x}, m_h) = (\vert m_{x1} - x_1 \vert) \lor (\vert m_{x2} - x_2 \vert) \lor \ldots \lor (\vert m_{xn} - x_n \vert)
\] (24)

On the other hand, when the med operator is used to generate the memory \( \mathbf{M} \), the region to which a feature vector \( \mathbf{x} \) belongs is determined using the median and the morphologic operator min.

\[
\text{index class} = \arg \left[ \bigwedge_{h} \left[ \bigvee_{j=1}^{n} \med m_{bj} - \med x_j \right] \right]
\]
Here, \[ \| \text{med} m_{hj} - \text{med} x_j \| \equiv d(x,m_h) \] is a metric that indicates the degree of belonging of the feature vector \( x \) to each one of the \( N \) classes.

Considering that \( m_{h1}, m_{h2}, ..., m_{hn} \) and \( x_1, x_2, ..., x_n \) are sets of finite values in either ascending or descending order, the median has two cases: if \( n \) is an odd number, then the median is the value defined as \( m_{h((n+1)/2)} \) and \( x_{(n+1)/2} \); if \( n \) is an even number, then the median is the value defined as \[ \left( m_{h(n/2)} + m_{h((n/2) + 1)} \right)/2 \] and \( \left( x_{n/2} + x_{(n/2)+1} \right)/2 \), then:

\[
\begin{align*}
d(x,m_h) &= \left| m_{h((n+1)/2)} - x_{(n+1)/2} \right| \\
d(x,m_h) &= \left( m_{h(n/2)} + m_{h((n/2) + 1)} \right)/2 - \left( x_{n/2} + x_{(n/2) + 1} \right)/2
\end{align*}
\] (25)

The computation of \( d(x,m_h) \) is simpler when the \text{prom} operator is used because the values of \( x \) and \( M_h \) have not to be ordered, this fact implies the highest processing speed. On the other hand, when the \text{med} operator is used, the extreme values of the set do not have important effects on the results.

The result of \( d(x,m_h) \) computation for three cases, \text{prom}, \text{med} and \text{pmed} operators, is a column vector where each element indicates the degree of belonging of the input vector, \( x \), with each one of the \( N \) classes

\[
\begin{bmatrix}
d(x,m_1) \\
d(x,m_2) \\
\vdots \\
d(x,m_N)
\end{bmatrix}
\] (26)

Finally, to establish which class belongs to a new input vector, the operator \text{min} is applied to the vector in the expression (26). The class is indicated by the index of the row of \( M \) that presents the nearest matching with the input vector \( x \).

\[
\text{index class} = \arg \min_h \left[ \bigwedge_{h=1}^N d(x,m_h) \right]
\] (27)

The VQ process finishes when all the inputs vectors (image blocks) have been replaced by the index class to which it belongs.

In most of the data processing, the proposed algorithm makes use of morphological operations that is, sums and comparisons. Therefore, the proposed algorithm offered a high processing speed. The Fig. 4 shows the structure of the associative memory that implements the high-speed search algorithm.
3.4 Complexity of the high-speed search algorithm

In this subsection, we analyze both the time and space complexity of the proposed algorithm. The algorithm complexity is measured by two parameters: the time complexity, or how many steps need the algorithm to solve the problem, and the space complexity, or how much memory it requires. To compute these parameters, we will use the pseudocodes of the prom, med and pmed operators that our algorithm uses to quantify a input vector (see Algorithm 1).

3.4.1 Time complexity

In order to measure the algorithm time complexity, we first obtain the run time based on the number of elementary operations (EO) that our proposal realizes to classify a input vector. For prom and pmed operators, we consider pseudocode from Algorithm 1(a), thus in the worst case, the conditions of lines 9 and 16 will always be true. Therefore, the lines 10, 17 and 18 will be executed in all iterations, and then the internal loops realize the following number of EO:

\[
\left( \sum_{i=1}^{n}(9+3) \right) + 3 = 12\left( \sum_{i=1}^{n}1 \right) + 3 = 12n + 3
\]

\[
\left( \sum_{i=1}^{N}(5+3) \right) + 3 = 8\left( \sum_{i=1}^{N}1 \right) + 3 = 8N + 3
\]
Algorithm 1. Pseudocodes of the high-speed search algorithm: (a) prom and pmed operators, (b) med operator.

The next loop will repeat $12n + 3$ EO at each iteration:

$$\sum_{i=1}^{N} (12n + 3) + 3 = \sum_{i=1}^{N} 12n + 6 + 3 = N(12n + 6) + 3 = 12Nn + 6N + 3$$

Thus, the equation (28) defines the total number of EO that the algorithm realizes.

$$T(n) = (12Nn + 6N + 3) + (8N + 3) + 1 = 12Nn + 14N + 7 \quad (28)$$

where $N$ is a codebook size and $n$ is a pattern dimension.

Now, for med operator, we consider the worst case of the pseudocode from Algorithm 1(b), thus, the conditions of lines 15 and 23 will always be true and the “while loop” will be executed $z$ times in each iteration. From Algorithm 1(b), we can distinguish 3 phases: "arranges the vector input elements", "compute the median" and “compute an element of M”.

The “arranges the vector input elements” phase realizes the following number of EO:

$$\sum_{i=1}^{n} (7 + 3) + 3 = 10 \sum_{i=1}^{n} 1 + 3 = 10n + 3, \text{ without "while loop"}$$
\[ \frac{n(n+1)}{2} = 6n(n+1) = 6n^2 + 6n, \text{ only the "while loop"} \]

The “compute the median” phase realizes 10 EO and the “compute an element of \( M \)” phase realizes the following number of EO:

\[
\left( \sum_{k=1}^{N} (7+3)+3 \right) + 1 = 10\sum_{k=1}^{N} 1 + 4 = 10N + 4
\]

Finally, the equation (29) defines the total number of EO that the algorithm realizes.

\[
T(n) = \left( \left( 10n + 3 \right) + \left( 6n^2 + 6n \right) \right) + \left( 10 \right) + \left( 10N + 4 \right) = 6n^2 + 16n + 10N + 17 \quad (29)
\]

Also, we analyze the number and type of arithmetical operations used by our algorithm during the VQ process of an image. Then, for \textbf{prom} and \textbf{pmed} operators, we consider the pseudocode from Algorithm 1(a); furthermore, we know that this process is applied to image of \( hi \times wi \) size. Thus, the number of operations that our algorithm, with \textbf{prom} or \textbf{pmed} operators, needs to quantify the image depends on the image size, the codebook size \( N \) and the pattern dimension \( n \)

\[
\left( \frac{hi \times wi}{n} \right) \left[ Nn(op2) + N(op1) \right] \quad (30)
\]

where \( op1 = 1 \) comparison and \( op2 = 1 \) sum and 1 comparison; \( \left( \frac{hi \times wi}{n} \right) \) is the number of patterns to quantify.

\( Nn(op2) \) is the number and type of arithmetical operations used to perform a morphological dilation over the absolute difference of the EAM-codebook elements and the input pattern components; \( N(op1) \) is the number and type of arithmetical operations used to perform a morphological erosion over the result of the previous process.

Now, for \textbf{med} operator, we consider pseudocode from Algorithm 1(b). Thus, the number of operations that our algorithm, with \textbf{med} operator, needs to quantify the image depends on the image size, the codebook size \( N \), the pattern dimension \( n \) and the computation of the median

\[
\left( \frac{hi \times wi}{n} \right) \left[ n(op1) + (op2) + N(op3) \right] \quad (31)
\]

where \( op1 = 2 \) sums and 2 comparisons, \( op2 = 2 \) sums, 1 comparison and 3 shift, \( op2 = 1 \) sum and 1 comparison. \( n(op1) \) is the number and type of arithmetical operations used to arranges the vector input elements; \( (op2) \) is the number and type of arithmetical operations used to compute the median and \( N(op3) \) is the number and type of arithmetical operations used to compute an element of \( M \).

### 3.4.2 Space complexity

The algorithm space complexity is determined by the amount of memory required for its execution. To quantify an image of \( hi \times wi \) size, which contains \( M = \left( \frac{hi \times wi}{n} \right)n \) \( n \)-dimensional
patterns, the proposed algorithm implementation for prom and pmed operators, requires two vectors \( \text{arg}[N] \) and \( \text{indexes}[M] \), and one matrix \( \text{prom\_codebook}[N \times n] \). Thus, the number of memory units (\( \mu \)) required for this process is:

\[
\mu_{\text{arg}} + \mu_{\text{indexes}} + \mu_{\text{prom\_codebook}} = N + M + N(n) = M + N(n+1)
\]

(32)

When the med operator is used, our algorithm only requires two vectors \( \text{indexes}[M] \) and \( \text{med\_codebook}[N] \). Thus, the number of \( \mu \) required is:

\[
\mu_{\text{indexes}} + \mu_{\text{med\_codebook}} = M + N
\]

(33)

For grayscale image (8 bits/pixel), the variables arg, prom_codebook and med_codebook are declared type byte, whereas the variable indexes depends on the codebook size, thus, if \( N \leq 256 \) then indexes is type byte and if \( N > 256 \) then indexes is type short (16 bit integer signed numbers).

Then the total number of bytes required by the implementation of the algorithm with prom or pmed operators is:

\[
M + N(n + 1) \quad \text{if} \quad N \leq 256
\]

\[
2M + N(n + 1) \quad \text{if} \quad N > 256
\]

(34)

and, the total number of bytes required by the implementation of the algorithm with med operator is:

\[
M + N \quad \text{if} \quad N \leq 256
\]

\[
2M + N \quad \text{if} \quad N > 256
\]

(35)

The number of memory units depends on the image size, codebook size and the codeword size chosen for the VQ process.

4. Experimental results

This section presents the experimental results obtained when our proposal is applied to an image quantization process. First, we show the proposed algorithm performance when it is using the prom, med and pmed operators. Then, we compare the proposed algorithm performance with the traditional LBG algorithm. The evaluated parameters are, the distortion generated on the reconstructed image and the influence that the VQ process has in the image compression when diverse codification methods are used. Finally, we analyze the number and type of operations and the amount of memory used by the proposed algorithm and the traditional LBG algorithm. For this purpose, a set of standard test images of size 512×512 pixels and 256 gray levels (Fig. 5) was used in simulations.

In order to measure the performance of both our proposal and LBG algorithm, we used a popular objective performance criterion named peak signal-to-noise ratio (PSNR), which is defined as
Fig. 5. Set of test images: (a) Lena, (b) Peppers, (c) Elaine, (d) Man, (e) Barbara, (f) Baboon.

\[ PSNR = 10 \log_{10} \left( \frac{\left(2^n - 1\right)^2}{\frac{1}{M} \sum_{i=1}^{M} (p_i - \tilde{p}_i)^2} \right) \]  

(36)

where \( n \) is the number of bits per pixel, \( M \) is the number of pixels in the image, \( p_i \) is the \( i \)-th pixel in the original image, and \( \tilde{p}_i \) is the \( i \)-th pixel in the reconstructed image.

The first experiment has the purpose to determine the performance that the algorithm proposed has with each of the operators (prom, med and pmed) when it is applied on the test images. The table 1 includes the distortion that each operator adds to the images during the quantization process when different sizes of codebook are used; the results show that the prom operator generates the minor distortion when the algorithm is applied on the test images.

In this experiment a codebook was generated for each test image to determine which image presents the best performance when it is used as training set; the best results were obtained when the image Lena was used to generate the EAM-codebook.

To minimize the distortion generated in the quantification process of images that were not used to obtain the EAM-codebook, it is possible to use an evolutionary method that allows adding information of new images to the book. This aspect is the subject of future work based on paper where the authors proposed the design of an evolutionary codebook using morphological associative memories (Guzmán et al., 2007).

The second experiment has the objective to compare the distortions that both the proposed algorithm (with prom operator) and LBG algorithm (the most widely quantization method used) add to the original image. For this purpose, the codebook was generated using the image Lena as the training set and the results were obtained using different sizes of the codebook. Then, VQ was applied to the set of the test images in order to determine the behavior of the algorithms with the patterns that do not belong to the training set. Table 2 shows the results of this experiment.

From Table 2, we can make the following observations: 1) the results obtained show that the proposed method is competitive with the LBG algorithm in the PSNR parameter; 2) the proposed algorithm replaced an input pattern by the index of the codeword that presents the nearest matching, not necessarily one of those already used to built the M memory. This property allows our algorithm to quantify efficiently the images that have not been used in the generation of the EAM-codebook.

In the third experiment, the performance of diverse standard coding methods applied to the quantified image with the proposed algorithm was evaluated. These methods included statistical modeling techniques, such as arithmetical, Huffman, range, Burrows Wheeler transformation, PPM, and dictionary techniques, LZ77 and LZP. The purpose of the third experiment is to analyze the proposed algorithm performance in image compression. For
this purpose, a coder that includes our algorithm and diverse entropy coding techniques was developed. Table 3 shows the compression results obtained from applying the coder on test images expressed as bit per pixel (bpp).

<table>
<thead>
<tr>
<th>Operator</th>
<th>Images</th>
<th>Proposed algorithm Codebook size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSNR</td>
</tr>
<tr>
<td>Lena 26.4166</td>
<td>27.4702</td>
<td>28.3959</td>
</tr>
<tr>
<td>Elaine 28.3715</td>
<td>29.1042</td>
<td>29.6899</td>
</tr>
<tr>
<td>Man 23.2950</td>
<td>23.9506</td>
<td>24.5434</td>
</tr>
<tr>
<td>Barbara 21.3870</td>
<td>21.7271</td>
<td>22.0764</td>
</tr>
<tr>
<td>Baboon 18.1937</td>
<td>18.5259</td>
<td>18.8603</td>
</tr>
</tbody>
</table>

Table 1. Proposed algorithm performance with prom, med and pmed operators (the image Lena was used to generate the codebook).

<table>
<thead>
<tr>
<th>Images</th>
<th>Proposed algorithm Codebook size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
</tr>
<tr>
<td>Lena 27.1448</td>
<td>28.2131</td>
</tr>
<tr>
<td>Peppers 26.3830</td>
<td>27.1805</td>
</tr>
<tr>
<td>Elaine 28.9191</td>
<td>29.6845</td>
</tr>
<tr>
<td>Barbara 21.8144</td>
<td>22.2457</td>
</tr>
<tr>
<td>Baboon 18.8927</td>
<td>19.3438</td>
</tr>
</tbody>
</table>

Table 2. Performance comparison of the proposed algorithm and the LBG algorithm.
Table 3. Compression results obtained from applying several entropy encoding technique on the information generated from the proposed algorithm.
<table>
<thead>
<tr>
<th>Images</th>
<th>Entropy encoding technique</th>
<th>Proposed algorithm (med)</th>
<th>Proposed algorithm (pmed)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Codebook size</td>
<td>Codebook size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bpp</td>
<td>bpp</td>
</tr>
<tr>
<td>Elaine</td>
<td>PPM</td>
<td>0.238</td>
<td>0.280</td>
</tr>
<tr>
<td></td>
<td>SZIP</td>
<td>0.254</td>
<td>0.297</td>
</tr>
<tr>
<td></td>
<td>Burrown</td>
<td>0.258</td>
<td>0.298</td>
</tr>
<tr>
<td></td>
<td>LZP</td>
<td>0.269</td>
<td>0.311</td>
</tr>
<tr>
<td></td>
<td>LZ77</td>
<td>0.289</td>
<td>0.332</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>0.324</td>
<td>0.360</td>
</tr>
<tr>
<td>Peppers</td>
<td>PPM</td>
<td>0.232</td>
<td>0.274</td>
</tr>
<tr>
<td></td>
<td>SZIP</td>
<td>0.245</td>
<td>0.287</td>
</tr>
<tr>
<td></td>
<td>Burrown</td>
<td>0.250</td>
<td>0.292</td>
</tr>
<tr>
<td></td>
<td>LZP</td>
<td>0.259</td>
<td>0.301</td>
</tr>
<tr>
<td></td>
<td>LZ77</td>
<td>0.275</td>
<td>0.320</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>0.327</td>
<td>0.362</td>
</tr>
<tr>
<td>Lena</td>
<td>PPM</td>
<td>0.254</td>
<td>0.300</td>
</tr>
<tr>
<td></td>
<td>SZIP</td>
<td>0.271</td>
<td>0.315</td>
</tr>
<tr>
<td></td>
<td>Burrown</td>
<td>0.277</td>
<td>0.320</td>
</tr>
<tr>
<td></td>
<td>LZP</td>
<td>0.283</td>
<td>0.327</td>
</tr>
<tr>
<td></td>
<td>LZ77</td>
<td>0.292</td>
<td>0.339</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>0.338</td>
<td>0.375</td>
</tr>
<tr>
<td>Barbara</td>
<td>PPM</td>
<td>0.266</td>
<td>0.310</td>
</tr>
<tr>
<td></td>
<td>SZIP</td>
<td>0.283</td>
<td>0.324</td>
</tr>
<tr>
<td></td>
<td>Burrown</td>
<td>0.285</td>
<td>0.328</td>
</tr>
<tr>
<td></td>
<td>LZP</td>
<td>0.299</td>
<td>0.341</td>
</tr>
<tr>
<td></td>
<td>LZ77</td>
<td>0.303</td>
<td>0.346</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>0.334</td>
<td>0.370</td>
</tr>
<tr>
<td>Man</td>
<td>PPM</td>
<td>0.261</td>
<td>0.307</td>
</tr>
<tr>
<td></td>
<td>SZIP</td>
<td>0.280</td>
<td>0.324</td>
</tr>
<tr>
<td></td>
<td>Burrown</td>
<td>0.279</td>
<td>0.323</td>
</tr>
<tr>
<td></td>
<td>LZP</td>
<td>0.300</td>
<td>0.344</td>
</tr>
<tr>
<td></td>
<td>LZ77</td>
<td>0.301</td>
<td>0.345</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>0.323</td>
<td>0.361</td>
</tr>
<tr>
<td>Baboon</td>
<td>PPM</td>
<td>0.313</td>
<td>0.353</td>
</tr>
<tr>
<td></td>
<td>SZIP</td>
<td>0.331</td>
<td>0.368</td>
</tr>
<tr>
<td></td>
<td>Burrown</td>
<td>0.405</td>
<td>0.367</td>
</tr>
<tr>
<td></td>
<td>LZP</td>
<td>0.350</td>
<td>0.391</td>
</tr>
<tr>
<td></td>
<td>LZ77</td>
<td>0.339</td>
<td>0.373</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>0.334</td>
<td>0.368</td>
</tr>
</tbody>
</table>

Table 3. Compression results obtained from applying several entropy encoding technique on the information generated from the proposed algorithm (continuation).
These results show that when these coding methods are applied on the results obtained by our algorithm, the entropy coding technique that offers the best results in compression ratio 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 based on the sequence of previous elements.

Furthermore, these results show that the proposed algorithm remains competitive with the algorithm LBG in the compression parameter.

Finally, based on results obtained in section 3.4 and the study for LBG algorithm presented in (Guzmán et al., 2008), a complexity analysis of both algorithms was realized. Table 4 summarizes computation complexities of the encoding phase of both LBG and proposed algorithm in terms of the type and average number of operations per pixels. This experiment was performed for \( n = 16 \) and 64, which are the most popular pattern dimensions. With regard to time complexity, this Table shows that the proposed algorithm provides considerable improvement over the LBG algorithm. Table 4 also shows that the memory required by our algorithm is smaller than the memory needed by the LBG when a VQ process is performed.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Pattern dimension ((n))</th>
<th>Codebook size ((N))</th>
<th>Required memory (bytes)</th>
<th>Average number of operations per pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CMP</td>
</tr>
<tr>
<td>LBG</td>
<td>16</td>
<td>128</td>
<td>41088</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td>49280</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>65664</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>128</td>
<td>41472</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td>74240</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>139776</td>
<td>8</td>
</tr>
<tr>
<td>Our proposal with prom and pméd operators</td>
<td>16</td>
<td>128</td>
<td>18560</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td>20736</td>
<td>272</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>41472</td>
<td>544</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>128</td>
<td>12416</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td>20736</td>
<td>260</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>37376</td>
<td>520</td>
</tr>
<tr>
<td>Our proposal with med operator</td>
<td>16</td>
<td>128</td>
<td>16512</td>
<td>10.062</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td>16640</td>
<td>18.062</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>33280</td>
<td>34.062</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>128</td>
<td>4224</td>
<td>4.015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td>4352</td>
<td>6.015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>8704</td>
<td>10.015</td>
</tr>
</tbody>
</table>

Table 4. Computation complexities of the encoding phase of both proposed algorithm and the LBG algorithm.
5. Conclusions

In the present work, we have proposed the use of extended associative memories in a high-speed search algorithm applied to image quantization. With the purpose of evaluating the influence that the proposed VQ algorithm has in the image compression, it was integrated in a codec where the codification stage includes several standard codification methods. The compression ratio and the signal to noise ratio obtained by our proposal show that the rate of compression and the decoding quality remain competitive with respect to the LBG algorithm.

Furthermore, the EAM has the property of substituting the input pattern by the class index that present the nearest match, without taking care that they have not been used in the construction of the associative memory. Our Algorithm inherited this property. This property allows our algorithm to quantify efficiently images that have not been used in the generation of the EAM-codebook, this is the reason it obtains good results in the decoding quality parameter.

Finally, in most of the data processing, the proposed algorithm makes use of the morphological operations, that is to say, its operation is based on maximums or minimums of sums, it uses only the operations of sums and comparisons. Therefore, the proposed algorithm offers a high processing speed in the search process of the encoding phase. Thus, with respect to traditional VQ algorithms, the main advantages offered by the proposed algorithm are, a high processing speed and low demand of resources (system memory).

For these reasons, we can conclude that our proposal provides a considerable improvement over the LBG algorithm.

6. References


Search algorithms aim to find solutions or objects with specified properties and constraints in a large solution search space or among a collection of objects. A solution can be a set of value assignments to variables that will satisfy the constraints or a sub-structure of a given discrete structure. In addition, there are search algorithms, mostly probabilistic, that are designed for the prospective quantum computer. This book demonstrates the wide applicability of search algorithms for the purpose of developing useful and practical solutions to problems that arise in a variety of problem domains. Although it is targeted to a wide group of readers: researchers, graduate students, and practitioners, it does not offer an exhaustive coverage of search algorithms and applications. The chapters are organized into three parts: Population-based and quantum search algorithms, Search algorithms for image and video processing, and Search algorithms for engineering applications.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:


InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821