Appl. Math. and Comp. Sci., 1996, vol.6, No.3, 431–445

IMAGE COMPRESSION BY COMPETITIVE LEARNING NEURAL NETWORKS AND PREDICTIVE VECTOR QUANTIZATION

LESZEK RUTKOWSKI*, ROBERT CIERNIAK*

This paper presents a combination of the vector quantization (VQ) technique with traditional (scalar) differential pulse code modulation (DPCM). A new algorithm for image compression, called predictive vector quantization (PVQ), is developed based on competitive neural networks and optimal linear predictors. The experimental results are presented and the performance of the algorithm is discussed.

1. Introduction

Image compression is an essential part of many applications such as high-definition television, video conferencing, facsimile transmission, image database, etc. A fundamental goal of image compression is to reduce the amount of data used to represent an image. Numerous compression techniques have been proposed. Most of them fall into two categories: predictive coders (see e.g. Hang and Woods, 1985) and transform coders (see e.g. Li, 1991). Recently, neural networks have emerged as a powerful tool for image compression (Ahalt *et al.*, 1990; Abbas and Fahmy, 1993; Fang *et al.*, 1992; Fowler *et al.*, 1993; Lu and Shin, 1992). In this paper, we combine the vector quantization (VQ) technique (Gray, 1984; Nasrabadi and King,1988) with traditional (scalar) differential pulse code modulation (DPCM). A new algorithm for image compression, named predictive vector quantization (PVQ), is developed based on competitive neural networks and optimal linear predictors. The experimental results show that our algorithm overperforms previous approaches to image compression.

2. Preliminaries

We assume that an image is represented by an $N_1 \times N_2$ array of pixels y_{ij}, $i = 1,2,\ldots,N_1$, $j = 1,2,\ldots,N_2$ (see Fig. 1). The image is partioned into contiguous small blocks of dimension $n_1 \times n_2$ (see Fig. 2).

$$\boldsymbol{Y}(m,n) = \begin{bmatrix} y_{1,1}(m,n) & \ldots & y_{1,n_2}(m,n) \\ \vdots & \ddots & \vdots \\ y_{n_1,1}(m,n) & \ldots & y_{n_1,n_2}(m,n) \end{bmatrix} \tag{1}$$

* Department of Electrical Engineering, Technical University of Częstochowa, 42–200 Częstochowa, Al. Armii Krajowej 17, Poland, e-mail: lrutko@zeia.el.pcz.czest.pl

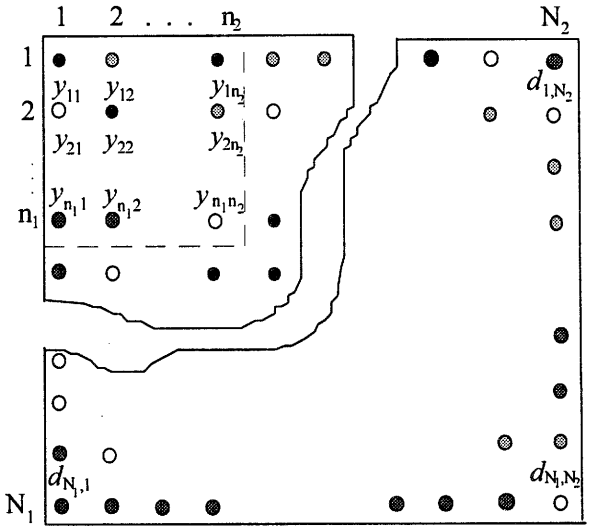

Fig. 1. Pixel representation of an image.

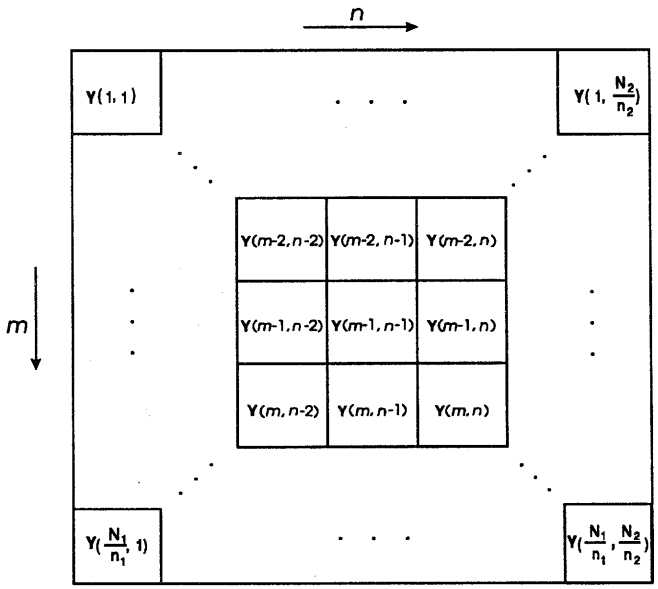

Fig. 2. Block representation of an image.

where $m = 1, 2, \ldots, N_1/n_1$ and $n = 1, 2, \ldots, N_2/n_2$. Obviously it is assumed that $\left(\frac{N_1}{n_1}\right) \bmod 1 = 0$ and $\left(\frac{N_2}{n_2}\right) \bmod 1 = 0$

In the sequel, the arrays (1) will be represented by the corresponding vectors

$$\boldsymbol{V}(m,n) = \left[v_1(m,n), v_2(m,n), \ldots, v_q(m,n)\right]^T \tag{2}$$

where we identify

$$v_1(m,n) = y_{1,1}(m,n), v_2(m,n) = y_{1,2}(m,n), \ldots, v_q(m,n) = y_{n_1,n_2}(m,n) \tag{3}$$

and $q = n_1 n_2$. This means that the original image is represented by $N_1 N_2/q$ vectors $\boldsymbol{V}(m,n)$.

3. General Architecture of the PVQ Algorithm

The general architecture of the predictive vector quantization algorithm (PVQ) is depicted in Fig. 3. This architecture is a straightforward vector extension of the traditional (scalar) differential pulse code modulation (DPCM) scheme (Gonzales and Woods, 1992; Jain, 1989).

The block diagram of the PVQ algorithm consists of an encoder and decoder, each containing an identical predictor, codebook and vector quantizer. The successive input vectors $\boldsymbol{V}(m,n)$ are introduced to the encoder and the difference $\boldsymbol{E}(m,n) = [e_1(m,n), e_2(m,n), \ldots, e_q(m,n)]^T$ given by

$$\boldsymbol{E}(m,n) = \boldsymbol{V}(m,n) - \overline{\boldsymbol{V}}(m,n) \tag{4}$$

is formed, where $\overline{\boldsymbol{V}}(m,n) = [\overline{v}_1(m,n), \overline{v}_2(m,n), \ldots, \overline{v}_q(m,n)]^T$ is the predictor of $\boldsymbol{V}(m,n)$. As in the scalar DPCM, the difference $\boldsymbol{E}(m,n)$ requires fewer quantization bits than the original subimage $\boldsymbol{V}(m,n)$. The next step is vector quantization of $\boldsymbol{E}(m,n)$. Mathematically, the vector quantization can be viewed as a mapping $\boldsymbol{VQ}$ from the q-dimensional Euclidean space $\mathbb{R}^q$ into a finite subset $\boldsymbol{G}$ of $\mathbb{R}^q$

$$\boldsymbol{VQ} : \mathbb{R}^q \longrightarrow \boldsymbol{G} \tag{5}$$

where

$$\boldsymbol{G} = \left[\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_J\right] \tag{6}$$

is the set of reproduction vectors (codewords or codevectors)

$$\boldsymbol{g}_j = \left[\boldsymbol{g}_{1,j}, \boldsymbol{g}_{2,j}, \ldots, \boldsymbol{g}_{q,j}\right]^T \tag{7}$$

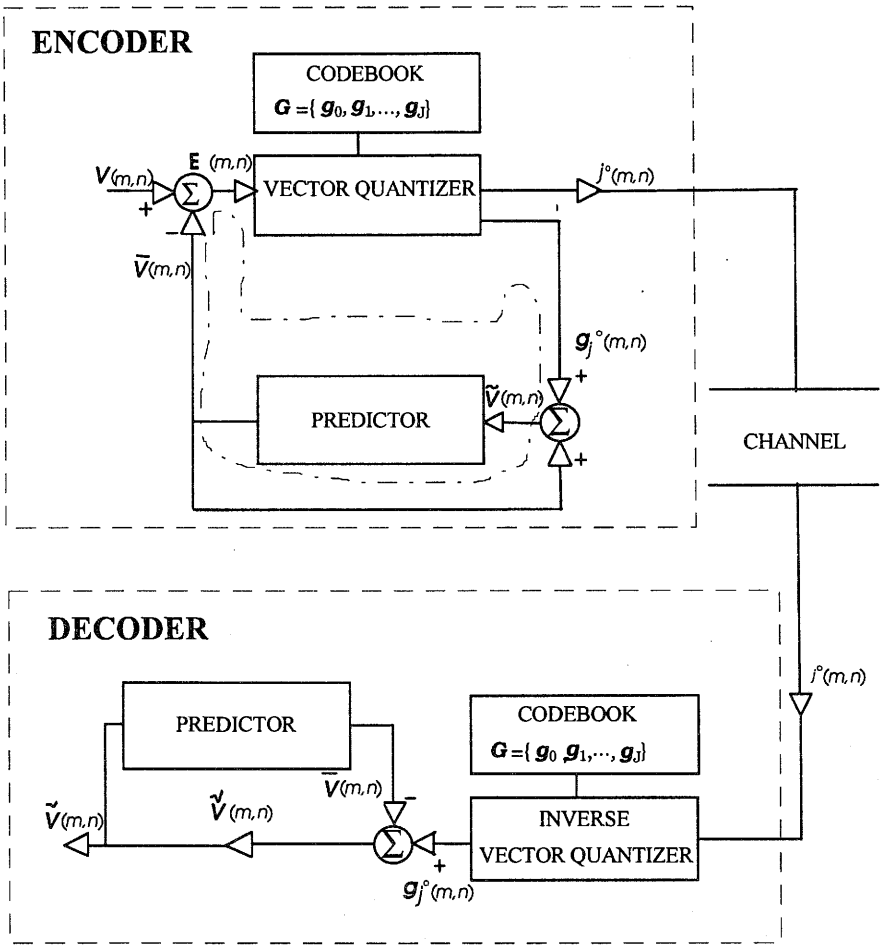

Fig. 3. Block diagram of the VQ DPCM image compression system.

The subset $\boldsymbol{G} \subset \mathbb{R}^q$ is commonly called the codebook. For every q-dimensional difference vector $\boldsymbol{E}(m,n)$, the distortion (usually the mean-squared error) between $\boldsymbol{E}(m,n)$ and every codeword $\boldsymbol{g}_j$, $j = 0,1,\ldots,J$ is computed. The codeword $\boldsymbol{g}_{j^0}(m,n)$ is selected as the representation vector for $\boldsymbol{E}(m,n)$ if

$$d_{j^0} = \min_{0 \le j \le J} d_j \tag{8}$$

where

$$d_j = \sqrt{\sum_{i=1}^{q} \left[e_i(m,n) - g_{ij} \right]^2} \tag{9}$$

The index $j^0(m,n)$ is broadcast via the transmission channel to the decoder. Mathematically, encoding is the mapping

$$\mathbb{R}^q \longrightarrow j^0 \tag{10}$$

In Section 5, we apply the competitive learning neural networks for construction of the vector quantizer. Observe that by adding the prediction vector $\overline{\boldsymbol{V}}(m,n)$ to the quantized difference vector $\boldsymbol{g}_{j^0}(m,n)$ we get the reconstructed approximation of the original input vector $\widetilde{\boldsymbol{V}}(m,n)$, i.e.

$$\widetilde{\boldsymbol{V}}(m,n) = \overline{\boldsymbol{V}}(m,n) + \boldsymbol{g}_{j^0}(m,n) \tag{11}$$

The fundamental theorem of the predictive vector quantization (Gersho and Gray, 1992) says that the overall reproduction error is equal to the error in quantizing the difference signal presented to the vector quantizer. The theorem follows from the simple fact that

$$\begin{aligned}\boldsymbol{V}(m,n)-\widetilde{\boldsymbol{V}}(m,n) &= \Big(\boldsymbol{V}(m,n)-\overline{\boldsymbol{V}}(m,n)\Big)-\Big(\widetilde{\boldsymbol{V}}(m,n)-\overline{\boldsymbol{V}}(m,n)\Big)\\ &= \boldsymbol{E}(m,n) - \boldsymbol{g}_j(m,n)\end{aligned} \tag{12}$$

As a measure of error between the original and reconstructed images one can take the mean-squared error

$$MSE = \frac{1}{N_1 N_2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\Big(y_{ij} - \widetilde{y}_{ij}\Big)^2 \tag{13}$$

or a signal-to-noise ratio (in decibel units)

$$SNR = 10\log_{10}\left(\frac{\Big(\max\{y_{ij}\}\Big)^2}{MSE}\right) \tag{14}$$

where $\widetilde{y}_{ij}$, $i = 1,2,\ldots,N_1$, $j = 1,2,\ldots,N_2$, stand for pixels of the reconstructed image.

The prediction vector $\overline{\boldsymbol{V}}(m,n)$ of the input vector $\boldsymbol{V}(m,n)$ is made from past observations of reconstructed vectors $\widetilde{\boldsymbol{V}}(m-k,n-l)$, $k = 1,2,\ldots,K$, and $l = 1,2,\ldots,L$. The predictor has the form

$$\overline{\boldsymbol{V}}(m,n) = \sum_{k=1}^{K}\sum_{l=1}^{L}\boldsymbol{A}_{kl}\widetilde{\boldsymbol{V}}(m-k,n-l) \tag{15}$$

where each $\boldsymbol{A}_{kl}$ is a $q \times q$ matrix, K and L are horizontal and vertical prediction orders, respectively. In the decoder, the index $j^0(m,n)$ transmitted by the channel is inverse vector-quantized

$$\boldsymbol{VQ}^{-1} : j^0 \longrightarrow \mathbb{R}^q \tag{16}$$

and the reconstructed vector $\widetilde{\boldsymbol{V}}(m,n)$ is formed in the same manner as in the encoder (see formula (11)).

The design of a predictive vector quantization scheme requires both a predictor (15) and a codebook (6) design. The simplest method to solve this problem is to use the open-loop design methodology suggested by Gersho and Gray (1992). This approach consists of two steps:

a) Design of the predictor based on the statistics of $\boldsymbol{V}(m,n)$, see Section 4.

b) Design of the codebook based on the ideal prediction residuals

$$\widehat{\boldsymbol{E}}(m,n) = \boldsymbol{V}(m,n) - \sum_{k=1}^{K}\sum_{l=1}^{L} \boldsymbol{A}_{kl}\boldsymbol{V}(m-k,n-1) \tag{17}$$

where $(k,l) \neq (0,0)$, see Section 5.

4. Vector Linear Prediction

Since the horizontal-vertical direction vector linear prediction seems to be rather complicated, we start with a simpler unidirectional problem. Let $\{\boldsymbol{X}(t)\}$ be a stationary random sequence of q-dimensional vectors with finite second moment, i.e. $E(\|\boldsymbol{X}(t)\|^2) < \infty$. The k-order predictor $\overline{\boldsymbol{X}}(t)$ of the current vector $\boldsymbol{X}(t)$ is given by

$$\overline{\boldsymbol{X}}(t) = \sum_{k=1}^{K} \boldsymbol{A}_k \boldsymbol{X}(t-k) \tag{18}$$

where $\boldsymbol{A}_k$ is a $q \times q$ prediction matrix. The prediction residual vector is defined as

$$\boldsymbol{e}(t) = \boldsymbol{X}(t) - \overline{\boldsymbol{X}}(t) \tag{19}$$

A vector predictor is said to be optimal if it minimizes the mean-squared error

$$D(t) = E\|\boldsymbol{X}(t) - \overline{\boldsymbol{X}}(t)\|^2 \tag{20}$$

The optimal coefficient matrices minimizing the error measure (20) satisfy the normal equation

$$\begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} & \cdots & \boldsymbol{R}_{1K} \\ \boldsymbol{R}_{21} & \boldsymbol{R}_{22} & \cdots & \boldsymbol{R}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{R}_{K1} & \boldsymbol{R}_{K2} & \cdots & \boldsymbol{R}_{KK} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_1^T \\ \boldsymbol{A}_2^T \\ \vdots \\ \boldsymbol{A}_K^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{10} \\ \boldsymbol{R}_{20} \\ \vdots \\ \boldsymbol{R}_{K0} \end{bmatrix} \tag{21}$$

or in the compact form

$$\boldsymbol{R}\boldsymbol{A} = \boldsymbol{r} \tag{22}$$

where

$$\boldsymbol{R}_{kl} = E\Big[\boldsymbol{X}(t-k)\boldsymbol{X}^T(t-l)\Big] \tag{23}$$

are the $K \times K$ correlation matrices, $k, l = 1, 2, \ldots, K$. The matrix $\boldsymbol{R}$ is a square $qK \times qK$ "supermatrix". The "supermatrix" $\boldsymbol{R}$ is a block-Toeplitz matrix. In the literature, eqn. (21) is called the multichannel normal equation (Kay, 1988; Marple, 1987) and can be solved recursively by making use of a generalized version of the Levinson-Durbin algorithm.

In practice, correlation matrices (23) are not known, but we have empirical data $\boldsymbol{X}(1), \boldsymbol{X}(2), \ldots, \boldsymbol{X}(N)$. The matrices $\boldsymbol{A}_k$ are estimated by minimizing the estimate of the mean-squared error (20)

$$\widehat{D}(t) = \sum_{t \in T} \left\| \boldsymbol{X}(t) - \overline{\boldsymbol{X}}(t) \right\|^2 = \sum_{t \in T} \left\| \boldsymbol{X}(t) - \sum_{k=1}^{K} \boldsymbol{A}_k \boldsymbol{X}(t-k) \right\|^2 \tag{24}$$

The estimates $\widehat{\boldsymbol{A}}_K$ of the matrices $\boldsymbol{A}_K$ satisfy the equation

$$\begin{bmatrix} \widehat{\boldsymbol{R}}_{11} & \widehat{\boldsymbol{R}}_{12} & \cdots & \widehat{\boldsymbol{R}}_{1K} \\ \widehat{\boldsymbol{R}}_{21} & \widehat{\boldsymbol{R}}_{22} & \cdots & \widehat{\boldsymbol{R}}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{\boldsymbol{R}}_{K1} & \widehat{\boldsymbol{R}}_{K2} & \cdots & \widehat{\boldsymbol{R}}_{KK} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{A}}_1^T \\ \widehat{\boldsymbol{A}}_2^T \\ \vdots \\ \widehat{\boldsymbol{A}}_K^T \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{R}}_{10} \\ \widehat{\boldsymbol{R}}_{20} \\ \vdots \\ \widehat{\boldsymbol{R}}_{K0} \end{bmatrix} \tag{25}$$

where

$$\widehat{\boldsymbol{R}}_{kl} = \sum_{t \in T} \left[\boldsymbol{X}(t-k) \boldsymbol{X}^T(t-l) \right] \tag{26}$$

According to the choice of the set T used in the empirical error measure (24), one can get different computational methods for estimation of correlation matrices. We consider two of such methods.

A. Autocorrelation Method

In this method, we use a rectangular window on the observed process and replace the true value of $\boldsymbol{X}(t)$ by zero for all values of t outside the observation interval $1 \leq t \leq N$, i.e.

$$\widetilde{\boldsymbol{X}}(t) = \boldsymbol{0} \qquad \text{for} \quad t \notin \{1, 2, \ldots, N\}$$

It is easily seen that the autocorrelation method leads to the following matrix equation:

$$\begin{bmatrix} \widehat{\boldsymbol{R}}_0 & \boldsymbol{R}_1 & \cdots & \widehat{\boldsymbol{R}}_{K-1} \\ \widehat{\boldsymbol{R}}_1 & \widehat{\widehat{\boldsymbol{R}}}_0 & \cdots & \widehat{\boldsymbol{R}}_{K-2} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{\boldsymbol{R}}_{K-1} & \widehat{\boldsymbol{R}}_{K-2} & \cdots & \widehat{\boldsymbol{R}}_0 \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{A}}_1^T \\ \widehat{\boldsymbol{A}}_2^T \\ \vdots \\ \widehat{\boldsymbol{A}}_K^T \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{R}}_1 \\ \widehat{\boldsymbol{R}}_2 \\ \vdots \\ \widehat{\boldsymbol{R}}_K \end{bmatrix} \tag{27}$$

where

$$\widehat{\boldsymbol{R}}_k = \sum_{t=1}^{N-k} \boldsymbol{X}(t)\boldsymbol{X}^T(t+k) \tag{28}$$

B. Covariance Method

In this method, we make no assumption about the data outside the interval $\{1,2,\ldots,N\}$. Now the range of values used for the error measure is $t \in \boldsymbol{T} = \{K, K+1,\ldots,N\}$. Consequently, the matrices $\widehat{\boldsymbol{R}}_{kl}$ in eqn. (25) take the form

$$\widehat{\boldsymbol{R}}_{kl} = \sum_{t=K}^{N} \boldsymbol{X}(t-k)\boldsymbol{X}^T(t-l) \tag{29}$$

The above methods can be extended to horizontal-vertical direction vector linear prediction. The normal equation takes the form

$$\sum_{k=0}^{K}\sum_{l=0}^{L} \boldsymbol{A}_{kl}\boldsymbol{R}_{kl,ij} = \boldsymbol{R}_{00,ij} \tag{30}$$

for $i = 0,1,\ldots,K,\ \ j = 0,1,\ldots,L,\ \ (l,k) \neq (0,0),\ \ (i,j) \neq (0,0)$, where

$$\boldsymbol{R}_{kl,ij} = E\Big[\boldsymbol{X}(m-k,n-l)\boldsymbol{X}^T(m-i,n-j)\Big] \tag{31}$$

$$\boldsymbol{R}_{00,ij} = E\Big[\boldsymbol{X}(m,n)\boldsymbol{X}^T(m-i,n-j)\Big] \tag{32}$$

If $K = 1$ and $L = 1$, then one gets

$$\begin{cases} \boldsymbol{A}_{10}\boldsymbol{R}_{10,10} + \boldsymbol{A}_{01}\boldsymbol{R}_{01,10} + \boldsymbol{A}_{11}\boldsymbol{R}_{11,10} = \boldsymbol{R}_{00,10} \\ \boldsymbol{A}_{10}\boldsymbol{R}_{10,01} + \boldsymbol{A}_{01}\boldsymbol{R}_{01,01} + \boldsymbol{A}_{11}\boldsymbol{R}_{11,01} = \boldsymbol{R}_{00,01} \\ \boldsymbol{A}_{10}\boldsymbol{R}_{10,11} + \boldsymbol{A}_{01}\boldsymbol{R}_{01,11} + \boldsymbol{A}_{11}\boldsymbol{R}_{11,11} = \boldsymbol{R}_{00,11} \end{cases} \tag{33}$$

In this case, the autocorrelation and covariance methods are also applicable with an obvious modification.

5. Neural Networks Techniques for Vector Quantization

In this section, we present competitive learning neural networks applied to vector quantization. Our goal is to find the codebook $\boldsymbol{G} = [\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_J]$ in order to minimize the performance measure

$$D = \sum_{m=1}^{N_1/n_1}\sum_{n=1}^{N_2/n_2} d^2\Big[\boldsymbol{E}(m,n), \boldsymbol{g}_{j^0}\Big] \tag{34}$$

where

$$d\Big[\boldsymbol{E}(m,n), \boldsymbol{g}_{j^0}\Big] = \min_{0\leq j\leq J}\Big\{d\Big[\boldsymbol{E}(m,n), \boldsymbol{g}_j\Big]\Big\} \tag{35}$$

and d is the distortion (usually chosen as the mean-squared error) between the vector $\boldsymbol{E}(m,n)$ and the codevector $\boldsymbol{g}_j$. The codevector $\boldsymbol{g}_{j^0}$ with minimum distortion is called the "winner". Figure 4 shows the competitive learning neural network.

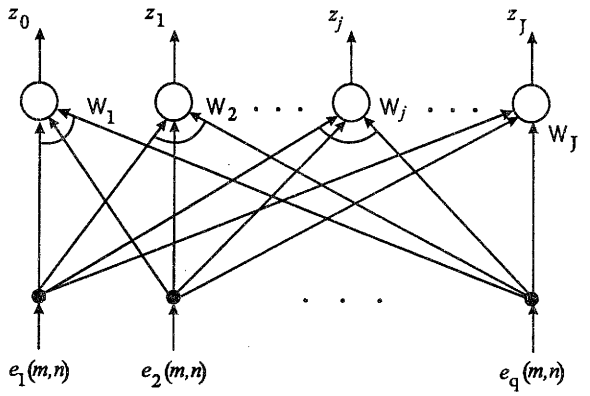

Fig. 4. Competitive learning neural network.

The elements of the input vector $\boldsymbol{E}(m,n) = [e_1(m,n), e_2(m,n), \ldots, e_q(m,n)]^T$ are connected to every neural unit having the weight $\boldsymbol{W}_j = [w_{1,j}, w_{2,j}, \ldots, w_{q,j}]^T$ and the output z_j, $j = 0, 1, \ldots, J$. The weights $\boldsymbol{W}_j$ are considered to be the codevectors, i.e.

$$\boldsymbol{G} = \left[\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_J\right] = \left[\boldsymbol{W}_0, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_J\right] \tag{36}$$

and the number of neural units $J+1$ is the size of the codebook.

We describe three alternative competitive learning neural networks:

a) The competitive learning (CL) networks;

b) The Kohonen self organizing feature map (KSFM);

c) The frequency-sensitive competitive learning (FSCL) networks.

A. CL Network

The distortion measure takes the form (Kohonen, 1988; Lu and Shin, 1992)

$$d\left[\boldsymbol{E}(m,n), \boldsymbol{W}_j\right] = \left\|\boldsymbol{E}(m,n) - \boldsymbol{W}_j\right\| = \sqrt{\sum_{i=1}^{q}[e_i(m,n) - w_{ij}]^2} \tag{37}$$

The index j^0 is selected such that

$$d\left[\boldsymbol{E}(m,n), \boldsymbol{W}_{j^0}\right] = \min_{0 \le j \le J}\left\{d\left[\boldsymbol{E}(m,n), \boldsymbol{W}_j\right]\right\} \tag{38}$$

The output z_j of each unit is computed as follows:

$$z_j = \begin{cases} 1 & \text{for } j = j^0 \\ 0 & \text{for } j \neq j^0 \end{cases} \tag{39}$$

The new weight vector is computed as

$$\boldsymbol{W}_j(t+1) = \boldsymbol{W}_j(t) + \alpha\Big[\boldsymbol{E}(m,n) - \boldsymbol{W}_j(t)\Big] z_j \tag{40}$$

where α is a learning parameter decreasing to zero as learning progresses. It should be emphasized that some of the CL neural units may be underutilized. This limitation sometimes leads to a rather high distortion rate.

B. KSFM Network

In the KSFM structure (Kohonen, 1988), each neural unit has an associated topological neighbourhood of other neural units. During the training process, the neural unit indexed by j^0, as well neural units within a specified neighbourhood $\aleph(j^0)$ of j^0 are updated:

$$\boldsymbol{W}_j(t+1) = \boldsymbol{W}_j(t) + \alpha\Big[\boldsymbol{E}(m,n) - \boldsymbol{W}_j(t)\Big], \quad j \in \aleph(j^0) \tag{41}$$

The KSFM network overcomes the problem of underutilized nodes of the CL network. On the other hand, it requires additional computation to calculate the neighbourhood of the unit j^0 and to update corresponding units.

C. FSCL Network

In the FSCL network (Ahalt *et al.*, 1990; Fowler *et al.*, 1993), the winning neural unit j^0 is selected based on the modified distortion measure

$$d\Big[\boldsymbol{E}(m,n), \boldsymbol{W}_j\Big] = F(f_j)\Big\|\boldsymbol{E}(m,n) - \boldsymbol{W}_j\Big\| = F(f_j)\sqrt{\sum_{i=1}^{q}[e_i(m,n) - w_{ij}]^2} \tag{42}$$

where F is a suitably chosen function of the counter f_j. The counter f_j counts how frequently the neural unit j is the "winner". The recursive procedure takes the form

$$\boldsymbol{W}_j(t+1) = \boldsymbol{W}_j(t) + H(f_j)\Big[\boldsymbol{E}(m,n) - \boldsymbol{W}_j(t)\Big] z_j \tag{43}$$

where H is another function of the counter f_j.

6. Experimental Results

The original tomographic image was scanned as shown in Fig. 5. The scanning assumed the frame of size $N_1 \times N_2 = 256 \times 256$ and 256 grey levels for each pixel.

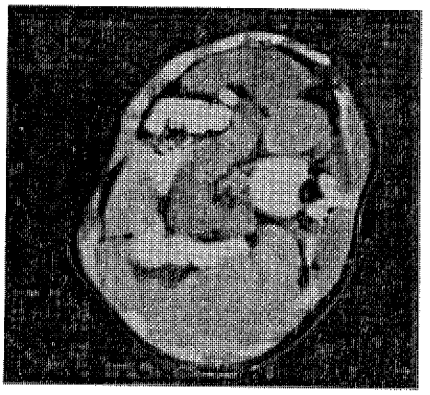

Fig. 5. The tested 256×256 pixels image.

The first experiment compares three competitive learning vector quantization techniques (CL, FSCL and KSFM) described in Section 5, under the following assumptions:

i) The first-order horizontal predictor $(K = 1,\ L = 0)$;

ii) The codebook size $J + 1 = 128$;

iii) The block subimage size $n_1 \times n_2 = 2 \times 2$.

Table 1 shows the SNR and MSE for three competitive learning neural networks.

Tab. 1. The SNR and MSE for three competitive learning neural networks.

Algorithm	SNR	MSE
CL	31.77	43.3
FSCL	35.02	20.47
KSFM	29.66	70.29

The reconstructed image and the difference between the original and reconstructed images are depicted in Fig. 6.

The experiments indicate that the FSCL method gives the best results. The following functions were selected in the FSCL algorithm:

$$F(f_j) = 1 - e^{-f_j/700}, \qquad H(f_{j^0}) = 0.1\, e^{-f_{j^0}/1000}$$

The next experiment shows the SNR and MSE for varying codebook sizes and the following functions F and H:

$$F(f_j) = e^{f_j/c}, \qquad H(f_{j^0}) = 0.1\, e^{-f_{j^0}/c}$$

The results are summarized in Table 2 for 3 epochs.

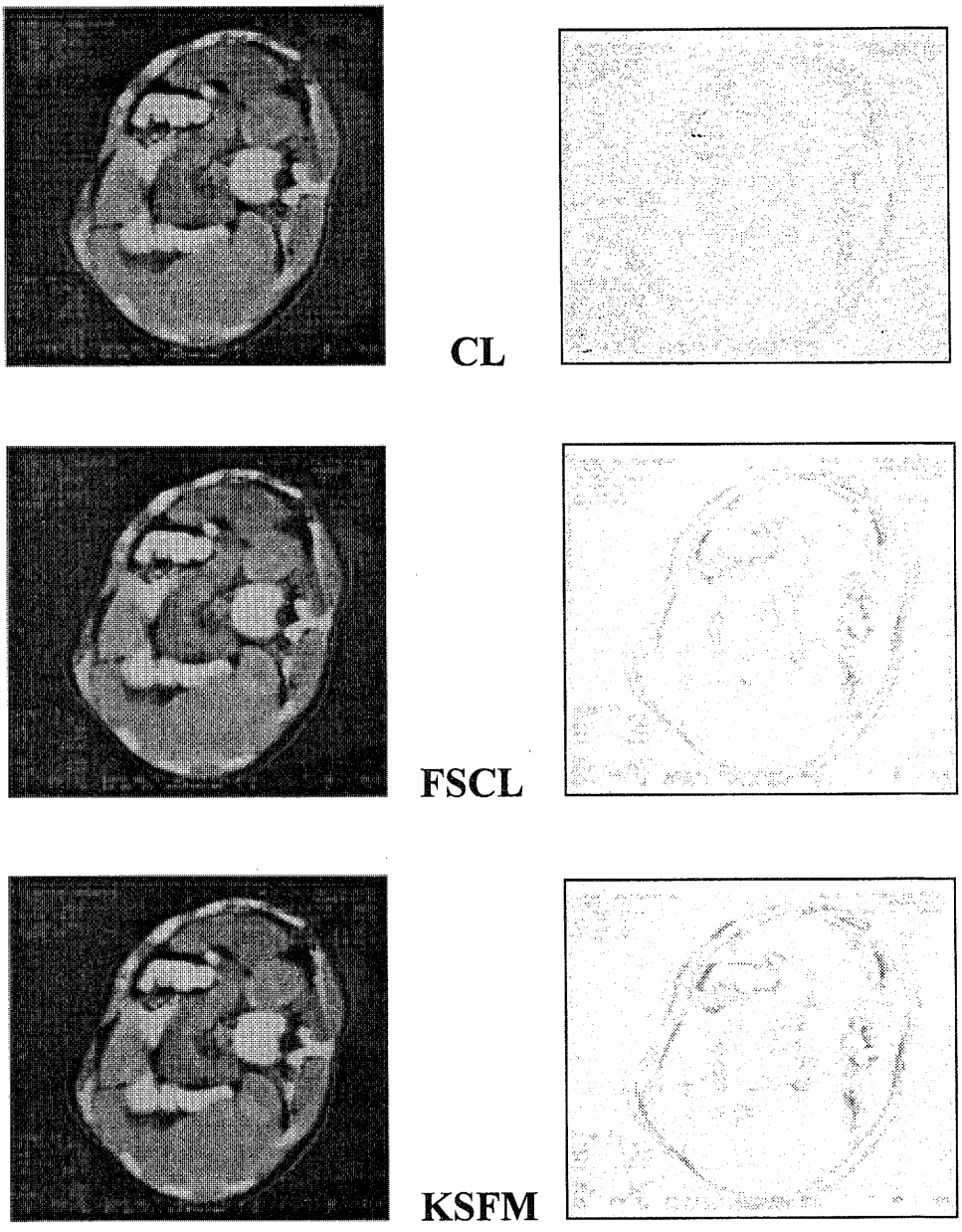

Fig. 6. Reconstructed image and difference between the original and reconstructed images.

Tab. 2. The SNR and MSE for a varying codebook size and parameter c.

$J+1$	c	SNR	MSE
64	100	31.03	51.3
64	500	31.25	48.8
64	700	32.83	33.9
64	1000	31.34	47.8
64	3000	33.30	30.4
128	100	32.83	33.9
128	700	34.65	22.3
128	1000	34.85	21.3

In Table 3 we present the SNR and MSE for a varying subimage size $n_1 \times n_2$ and different types of function F.

Tab. 3. The SNR and MSE for a varying subimage size $n_1 \times n_2$ and different types of functions F.

$n_1 \times n_2$	$F(f_j)$	$H(f_{j0})$	SNR	MSE
2×2	$1 - e^{-f_j/700}$	$0.1\, e^{-j_{j0}/1000}$	35.02	20.47
2×2	$e^{f_j/1000}$	$0.1\, e^{-j_{j0}/1000}$	34.85	21.28
4×4	$1 - e^{-f_j/700}$	$0.1\, e^{-j_{j0}/1000}$	28.53	91.26
4×4	$e^{f_j/1000}$	$0.1\, e^{-j_{j0}/1000}$	26.79	136.03
2×2	f_j	$0.1\, e^{-j_{j0}/1000}$	9.98	6527.14

Finally, we compare our method with previous approaches to "Lena" image compression. The results are presented in Table 4.

Tab. 4. Comparison of image compression techniques.

Method	Parameters	SNR	MSE	Compresion ratio
This article	$n_1 \times n_2 = 4 \times 4$ $J+1 = 128$	29.40	74.28	18.28:1
Fowler *et al.* (1993)	$n_1 \times n_2 = 4 \times 4$ $J+1 = 128$	24.42	234.99	18.28:1
Manikopoulous (1992)	non-linear predictor	29.5	72.62	15.6:1

It is easily seen that our method overperforms the previous PVQ approach (Fowler *et al.*, 1993) to image compression and is comparable to the nonlinear predictor approach (Manikopoulous, 1992). The performance of our method is depicted in Fig. 7.

a)

b)

c)

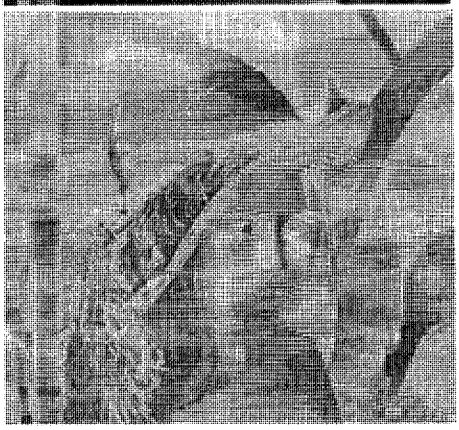

Fig. 7. (a) "Lena" original image.
(b) "Lena" reconstructed image.
(c) Difference between the reconstructed and original images.

7. Final Remarks

We investigated the predictive vector quantization algorithm for image compression. The codebook was design based on the competitive learning neural networks including the FSCL method giving the best result. Contrary to the previous similar approach (Fowler *et al.*, 1993) our predictor is chosen in an optimal way, which results in a better compression quality.

References

Abbas H.M. and Fahmy M.M. (1993): *Neural model for Karhunen-Loeve transform with application to adaptive image compression.* — IEE Proc.-I, v.140, No.2, pp.135–143.

Ahalt S.C., Krishnamurthy A.K., Chen P. and Melton D.E. (1990): *Competitive learning algorithms for vector quantization.* — Neural Networks, v.3, pp.277–290.

Fang W.C., Sheu B.J., Chen O.T.-C and Choi J. (1992): *A VLSI neural processor for image data compression using self-organization networks.* — IEEE Trans. Neural Networks, v.3, No.3, pp.506–517.

Fowler J.E., Carbonara M.R. and Ahalt S.C. (1993): *Image coding using differential vector quantization.* — IEEE Trans. Circuits and Systems for Video Technology, v.3, No.1, pp.350–367.

Gersho A. and Gray R.M. (1992): *Vector Quantization and Signal Compression.* — Boston-Dordrecht-London: Kluwer.

Gonzales R.C. and Woods R.E. (1992): *Digital Image Processing.* — Reading, Massachusets: Addison-Wesley.

Gray R. (1984): *Vector quantization.* — IEEE ASSP Magazine, v.1, No.2, pp.4–29.

Hang H.M. and Woods J.W. (1985): *Predictive vector quantization of images.* — IEEE Trans. Communications, v.33, No.11, pp.1208–1219.

Jain A.K. (1989): *Fundamentals of Digital Image Processing.* — Englewood Cliffs: Prentice-Hall.

Kay S.A. (1988): *Modern Spectral Estimation.* — Englewood Cliffs: Prentice Hall.

Kohonen T. (1988): *Self-Organization and Assotiative Memory.* — Berlin: Springer-Verlag.

Li W. (1991): *Vector transform and image coding.* — IEEE Trans. Circuits and Systems for Video Technology, v.1, No.4, pp.297–307.

Lu C.C. and Shin Y.H. (1992): *Neural networks for classified vector quantization of images.* — Engng. Applic. Artif. Intell., v.5, No.5, pp.451–456.

Manikopoulos C.N. (1992): *Neural networks approach to DPCM system design for image coding.* — IEE Proc.-I, v.139, No.5.

Marple S.L. (1987): *Digital Spectral Analysis with Application.* — Englewood Cliffs: Prentice Hall.

Nasrabadi N.M. and King R.A. (1988): *Image coding using vector quantization: A review.* — IEEE Trans. Communications, v.36, No.8, pp.957–971.