Systems Analysis Modelling Simulation
Vol. 43, No. 11, November 2003, pp. 1529–1543

IMAGE COMPRESSION USING SELF-ORGANIZING MAPS

C. AMERIJCKX*, J.-D. LEGAT† and M. VERLEYSEN‡

Microelectronics Laboratory, Université catholique de Louvain, 3 place du Levant, B-1348 Louvain-la-Neuve, Belgium

(Received 7 March 2002)

The self-organizing Kohonen map is a reliable and efficient way to achieve vector quantization. Typical application of such algorithm is image compression. Moreover, Kohonen networks realize a mapping between an input and an output space that preserves topology. This feature can be used to build new compression schemes which allow to obtain better compression rate than with classical method as JPEG without reducing the image quality. Compared to JPEG, our lossy compression scheme shows better performances (in terms of PSNR) for compression rates higher than 30 [C. Amerijckx, M. Verleysen, P. Thissen and J.-D. Legat (1998). Image compression by self-organized Kohonen map. *IEEE Transactions on Neural Networks*, **9**(3), 503–507.]. For lossless compression, this rate is about 2.7 for standard images.

Keywords: Image compression; Self-organizing maps; Kohonen maps; Vector quantization

1. INTRODUCTION

In the context of image processing, compression schemes are aimed to reduce the transmission rate for still (or fixed images), while maintaining a good level of visual quality.

Several compression techniques have been developed, such as DPCM (Differential Pulse Code Modulation), DCT (Discrete Cosine Transform) [2,3] and DFT (Discrete Fourier Transform), together with numerous vector quantization methods [4,5]. Vector quantization is based on the creation of a codebook, constituted by M elements (codewords). Each codeword is supported by a vector of the same size as the size of the input vectors. Each input vector will be approximated by a codeword in such a way that the global distortion will be minimized. In other words, the goal of the vector quantizer is to project the input space on a discrete one of the same dimension, constituted by M elements, while minimizing a distortion error function. On a reduced bandwidth transmission line, only the indexes of the successive codewords that approximate the best of the input vector will be transmitted, instead of the whole vector itself. A vector

*Present address: Alcatel Microelectronics, Excelsiorlaan 44, B-1930 Zaventem, Belgium.
†E-mail: legat@dice.ucl.ac.be
‡Corresponding author. E-mail: verleysen@dice.ucl.ac.be

ISSN 0232-9298 print: ISSN 1029-4902 online © 2003 Taylor & Francis Ltd
DOI: 10.1080/0232929032000115182

quantizer is thus characterized by the following relation:

$$Q : R^N \to Y|X \to \hat{X}_i, \quad \hat{X}_i \in Y, \ 1 \leq i \leq M, \tag{1}$$

where $X \in R^N$ is an input vector, Y is the codebook, $\hat{X}_i$ are the codewords ($1 \leq i \leq M$), and N is the dimension of the space. Each vector X will be projected on the nearest codeword, for a defined distance measure (here the Euclidean distance):

$$d(X, \hat{X}_i) = \min(d(X, \hat{X}_j)), \qquad 1 \leq j \leq M. \tag{2}$$

The best quantizer Q is the one which minimizes the distortion error

$$\mathrm{Err} = E(d(X, \hat{X}_i)) \tag{3}$$

over the whole input space.

For a few years, research has focused on artificial neural network methods for vector quantization; Kohonen's algorithm, or Kohonen's topological self-organizing map [6,7], is one of these methods that can be used for vector quantization. One function of this algorithm is to realize the quantization of a continuous input space of stimuli into a discrete output space, the codebook. Besides this vector quantization feature, the Kohonen algorithm has another essential property: self-organization. In fact, this algorithm is based on the observation of the brain that realizes self-organization of cartographies during the first stages of its development, starting with tactile, visual, or acoustic sensory inputs. Considering the example of sounds, the consequence of self-organization is that two sounds with neighboring frequencies excite two neighboring neurons of the brain. In terms of neural computation, this relation becomes: "two neighboring input vectors will excite two neighboring output vectors (codewords)," in the sense of a given distance function.

Other authors have already used Kohonen's neural network in image compression for its vector quantization property, in the same way as other vector quantizers, such as the LBG (Linde–Buzo–Gray algorithm) [8]. It was proved that both give similar results when considering the quality of the compressed image [9]; their compression rate is only dependent on the codebook size. Kohonen's network also accelerates the convergence of the quantization process [10]. In [11], the topology-preserving property is used for progressive transmission of the image. In [12], it is used for further differential coding, as in this chapter, but on images without DCT, and with a less-performant zeroth-order predictor (instead of first-order in our compression scheme).

We propose in this article to use the second property of Kohonen's neural network, self-organization, in order to achieve higher compression rates without any further decrease in the image quality. First, we describe each step of the compression scheme, and in particular the differential coding, which is a coding technique that allows to make the ordering (self-organization) property very efficient. Then, results of simulations are shown for image compression with this new method, including comparison with other compression schemes. Both lossy and lossless compression are considered.

2. THE GLOBAL COMPRESSION SCHEME

The global compression scheme for lossy compression is described in Fig. 1. After a vectorization (transformation of image blocks into vectors), a DCT [2] and a low-pass

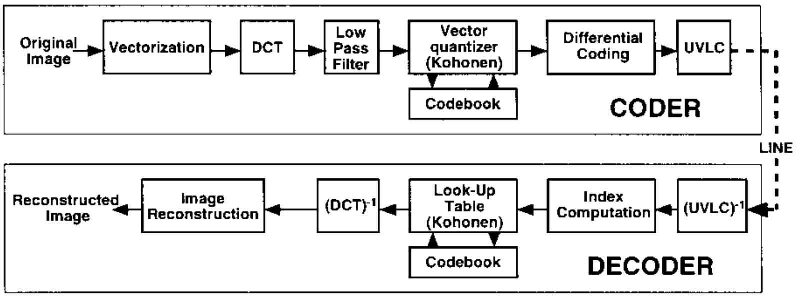

FIGURE 1 Global compression scheme for lossy compression.

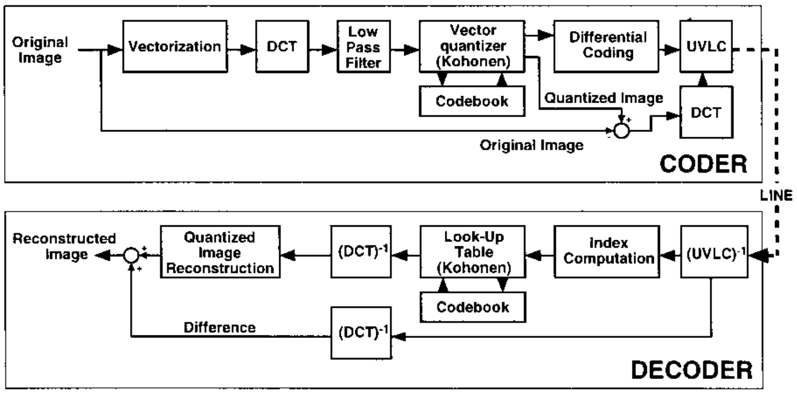

FIGURE 2 Global compression scheme for lossless compression.

filter first reduce the quantity of information by keeping only the low-frequency coefficients. Then, the vector quantization is performed, with another loss of information. Finally, the indexes of the codewords found by the vector quantizer are transformed by a differential coding, and the results are compressed by an entropic coder, Universal Variable Length Coder (UVLC) [13,14]; these last two steps do not introduce any loss in the information. The decompression scheme performs the same operations in the opposite way.

The global compression scheme for lossless compression is illustrated in Fig. 2. It is similar to the lossy scheme, except that the difference between the quantized image (after the Kohonen vector quantizer) and the original one is computed, and transmitted after being compressed by the entropic coder. In this way, the exact reconstruction of the original image is made possible.

2.1. Image Preprocessing

In order to use vector quantization techniques, the image has to be coded into vectors. Each of these vectors is then approximated and replaced by the index of one of the elements of the codebook: the only useful information in this index, for each of the blocks of the original image, reducing by this way the transmission bit rate.

The image is first decomposed into blocks (4×4 or 8×8 pixels as usual); the DCT transform is applied on each block, in order to eliminate a part of the information contained in the image, that is, high frequencies not visible to human eyes.

The DCT transform of a $k \times k$ pixels block is again a $k \times k$ block. However, in the transformed block, low-frequency coefficients are grouped in the upper-left corner, while high-frequency ones are grouped in the lower-right corner. The low-pass filter on the transformed block will thus keep only the l coefficients nearest from the upper left corner, with $l \leq k^2$; the remaining $k^2 - l$ coefficients are discarded, supposing that they do not contribute too much to the visual quality of the image. After vector quantization, we will thus have the twofold properties:

- for a given size of the codebook, the low-pass transformation allows a better subsequent quantization because it reduces the dimension of the space (l instead of k^2);
- for a given visual image quality, it allows to have a better compression rate by decreasing the codebook size.

2.2. Kohonen's Self-organizing Maps

As mentioned in the introduction, the goal of Kohonen's map is to create a correspondence between the input space of stimuli and the output space of codebook elements, the codewords or neurons. These have to approximate at best all vectors of the input space after the learning phase [6].

All neurons, or codewords, are physically arranged on a square grid; it is thus possible to define k-neighborhoods on the grid, which include all neurons whose distance (on the grid) from one (central) neuron is less than or equal to k.

Each of the M codewords is represented by its weight $\hat{X}_j \in R^N$. For each presentation of an input vector X during a training phase, the index i of the codeword nearest from X is determined, according to the Euclidean distance:

$$d(X, \hat{X}_i) = \min(d(X, \hat{X}_j)), \quad 1 \leq j \leq M \tag{4}$$

The selected neuron i, and all neurons in a k-neighborhood of neuron i, are then "moved" in the direction of the input vector X according to:

$$\hat{X}_m(t+1) = \hat{X}_m(t) + \alpha(t)(X - \hat{X}_m), \tag{5}$$

where m represents the index of all neurons in the k-neighborhood of neuron i and $\alpha(t)$ the learning factor. $\alpha(t)$ and k must decrease during the learning to ensure a good convergence of the algorithm.

During the learning phase, the network organizes itself with respect to the density of the presented vectors. After learning, two neurons physically neighbored on the grid, will have similar weights, leading to the well-known "organization" property of Kohonen networks.

This property is the key point of the compression scheme proposed in this chapter. Two successive blocks in the image will be coded, after the DCT transform, by similar vectors, since it is supposed that sharp variations in the image do not occur too often [15,16]. The vector quantization with the Kohonen algorithm will thus lead to two codewords located near together on the grid, because of the organization property of the network. The differential coding will then use this property to not to code the difference between the codewords themselves, but the difference between their indexes, difference which will be small in average too.

2.3. Differential Coding

Once the codebook has been constituted during the learning phase with the Kohonen algorithm, the next step is to code the differences between the indexes of the codewords. The differential coding makes it possible to use the topological ordering property of codebook created with Kohonen's algorithm.

Without using the organization property of Kohonen maps, quantization means to select the neuron nearest to the presented vector, and to send its index on the transmission line. So, the image is compressed if the number of bits necessary to send an index is lower than the number of bits used to code each pixel in a block. The ordering property of codebooks created with Kohonen's map makes it possible to be more efficient. Due to the fact that two neighboring blocks of the image are very often similar, because it is supposed that any sharp variations in an image do not occur often, the vectors corresponding to these two blocks are close to each other (when considering a given distance function). So, because of the organization of the network, two neighboring vectors in the input space will activate two neighboring neurons of the network. The two correspondent indexes will be close too, and their difference will be low in average. We will see in the next section that small values provided by the differential coder (small differences of indexes) will be coded more efficiently by the UVLC entropic coder; the goal here is thus to obtain differences between indexes as small as possible.

To compute differences between indexes, one has to determine which block will be taken as the "previous" one, when a new block is encoded. Two possibilities are presented in the following, the first one being very simple, the second one being more efficient.

2.3.1. *Using the Previous Encoded Block*

This method is the simplest one, and necessitates minimum number of computations. Since two consecutive blocks in an image are supposed to be very similar, the difference between the corresponding indexes of the respective neurons will be small, and this difference, without any further processing, will be transmitted to the UVLC entropic coder. Figure 3 shows the succession of the differences computations used in this method.

2.3.2. *Searching for the Best Minimum Gradient Direction in the Image*

The second method is based on the following principle: we suppose that gradients in the image are smooth, and thus that the direction in which the differences between two successive blocks was minimum for already encoded blocks will be the same as the direction in which the difference is minimum for the new block to encode.

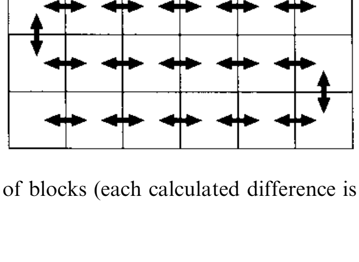

FIGURE 3 Succession of blocks (each calculated difference is represented by an arrow).

In other terms, and with the notations of Fig. 4, we suppose that the minimum difference between blocks *i* and *b*, *i* and *d*, *i* and *f*, and *i* and *h*, will be respectively in the same direction (*D*1, *D*2, *D*3, or *D*4) as the minimum difference between already encoded blocks *b* and *a*, *d* and *c*, *f* and *e*, and *h* and *g*. The advantage of this method is that the selected difference to code (between the four possible differences) will be always smaller (or equal) than the difference computed by the first method; the selected direction does not have to be coded: since blocks *a*–*h* have already been transmitted when coding block *i*, the direction of the minimal difference between them is known, and does not have to be transmitted again.

The coding of a block is summarized in Fig. 5. The best direction is first computed according to the above scheme, and block *b*, *d*, *f*, or *h* is selected. New block *i* is then encoded by the Kohonen's algorithm, and the difference between the two indexes is sent to the entropic coder.

The decoding is illustrated in Fig. 6. The best direction is first computed with the already decoded blocks; the sum of the selected index and of the transmitted difference

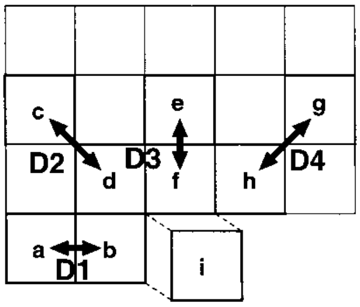

FIGURE 4 Determination of the best direction to calculate the difference between blocks.

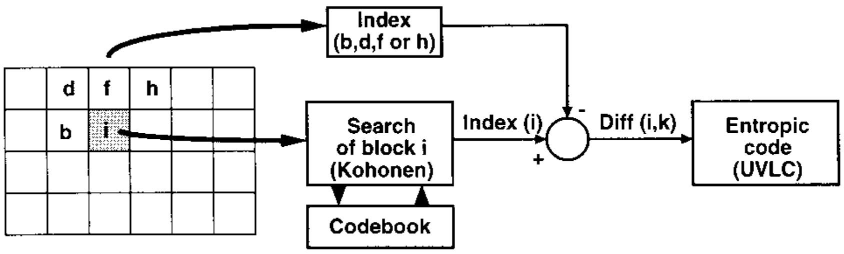

FIGURE 5 Coding of the difference between indexes.

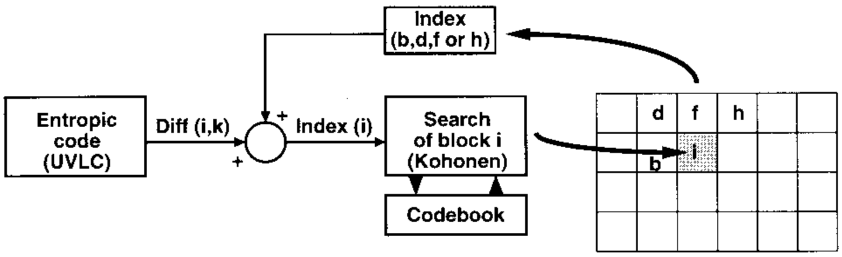

FIGURE 6 Decoding the difference between indexes.

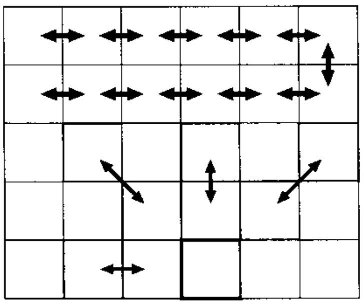

FIGURE 7 Complete differences scheme. Large arrows indicate the difference of blocks to transmit to the UVLC, and thin arrows indicate how to choose the best direction to use for coding, beginning at the third line.

is then the index of the new block *i* which is decoded, and converted again into an image block by the look-up table created with Kohonen's algorithm.

It may be mentioned that the scheme proposed in Fig. 4 is only valid when the first two lines have already been encoded. For the first two lines, the first proposed differential scheme is used, as illustrated in Fig. 7.

2.4. Entropic Coder: UVLC

Now the differences between indexes are computed, they have to be transformed into the smallest possible univoque binary sequence to be transmitted on the transmission line.

The UVLC is well appropriate to code small numbers. It is a universal entropic coder based on a variable length coding [13,14]. It minimizes the necessary information to code a series of small numbers by optimizing the length of the binary sequence used to transmit the run-lengths, that is, series of consecutive 0s on a line.

The principle is to concentrate the maximum number of 0s in order to optimize the length of the code generated by the UVLC. To achieve this, all vectors (corresponding to the differences of indexes generated as described above) are placed as *columns* in a matrix; since many of these vectors are small, the upper part of the matrix will contain many 0s, and the UVLC will then code this matrix *line* by *line*. For each line *i*, the following operations are performed (a run-length is defined as the number of consecutive 0s):

- the number and the length of all run-lengths are determined, as well as an optimal number m_i, that minimizes the length of the binary code needed to transmit all the run-lengths of this line (see [14] for more details);
- this number m_i is coded over a predefined number of bits;
- the line itself is coded in the following way:
    - the length RL of each run-length is determined;
    - while $\mathrm{RL} \geq 2^{m_i}$, send a 0 and $\mathrm{RL} = \mathrm{RL} - 2^{m_i}$;
    - when $\mathrm{RL} < 2^{m_i}$, send a 1 and RL coded over m_i bits;
    - the next bit of the line (after a run-length) is necessarily a 1; the whole *column* of the matrix below this 1 is then transmitted uncoded, and a flag is memorized for this column, indicating that it does not have to be regarded in subsequent lines;
- all sign bits of all columns are transmitted uncoded.

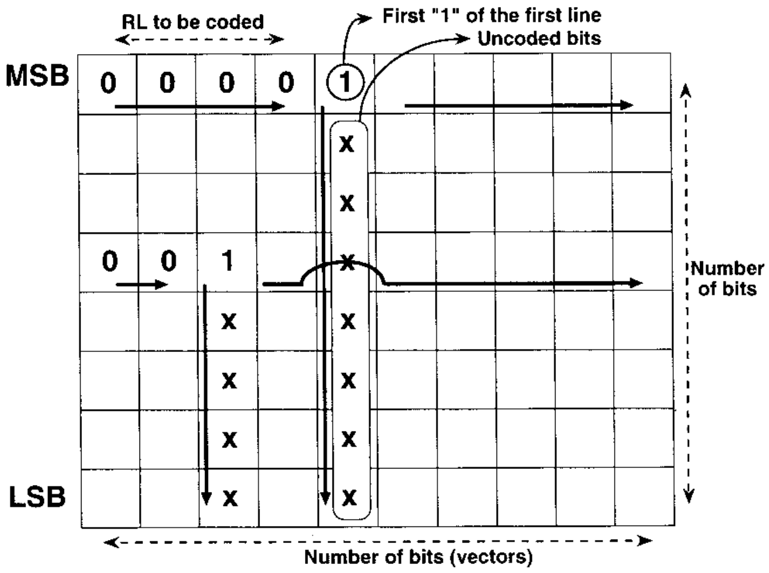

FIGURE 8 Example of UVLC coding on 10 vectors in 8 dimensions.

An example of UVLC coding is illustrated in Fig. 8. In the first line, a first run-length of four bits is coded (see above). Then, the whole fifth column is transmitted uncoded. The first line is then continued, in coding the next run-length,.... Column 5 is discarded for line 2 and the subsequent ones. In the fourth line for example, the first run-length of 2 bits is coded, then the remaining of the third column is sent uncoded, and the fourth line, beginning at the fourth column, is coded according to the same principle (the fifth column is no more considered). The process is continued while the code for a line is smaller than the number of bits of the line itself. The UVLC has the advantage (over conventional Hufmann's code for example) that it does not require any predefined table, and adapts itself to the input distribution.

3. SIMULATIONS AND RESULTS

3.1. Lossy Compression

For the simulation of the whole compression process (lossy scheme, see Fig. 1), we used 4×4 points image blocks (as a compromise between the compression rate and the correlation between successive blocks. The Kohonen algorithm was trained with an exponentially decreasing function $\alpha(t)$. While the simulations have been carried out on different images with similar results, the standard Lena image is used in this article for illustration purposes.

First, we have to show the consequences of the low-pass filtering: by removing a part of the high frequencies, we delete a part of the information contained in the image. An immediate consequence of this is a reduction of the peak signal to noise ratio (PSNR), even though the image visual quality remains more or less unchanged. In other words, before beginning the compression, the image quality will be degraded by the filtering. We can see in Table I the evolution of the PSNR on the Lena image, when the number of DCT coefficients kept after the low-pass filtering varies from 2 to 8.

To keep an acceptable subjective quality of the image, 6 and 8 DCT coefficients will be kept in the next simulations (PSNR of around 30 dB before vector quantization).

The tests made with our compression scheme consist of a variation of the codebook size for two given cut frequencies (6 and 8 DCT coefficients). The compression rate obtained in function of the codebook size (differential coding) is shown in Fig. 9 while their associated PSNR is given in Fig. 10. As a comparison, the compression rate without differential coding is equal to $CR = (16 \times 8)/Nbr$, where each bit of the image is coded on 8 bits, and Nbr is the number of bits necessary to code all codewords (7 for $128 = 2^7$ codewords, 8 for 256, and 9 for 512). Both simulations were made with the second proposed differential coding scheme (the selection of the best direction to code the differences).

TABLE I PSNR (in dB) after the low-pass filtering

	DCT2	*DCT3*	*DCT4*	*DCT5*	*DCT6*	*DCT7*	*DCT8*
Lena	25.09	26.54	26.83	27.87	29.35	30.07	31.15

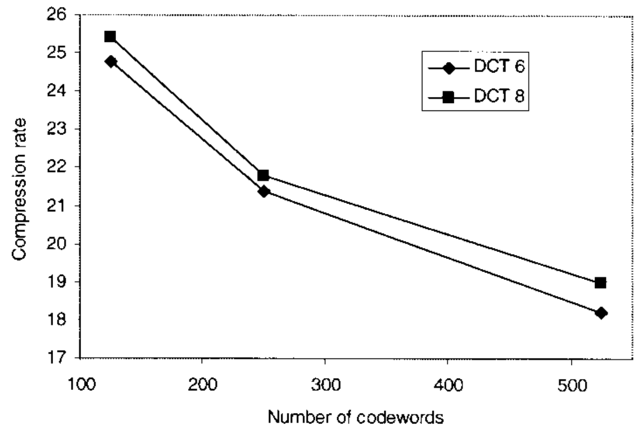

FIGURE 9 Compression rate evolution with the number of codewords.

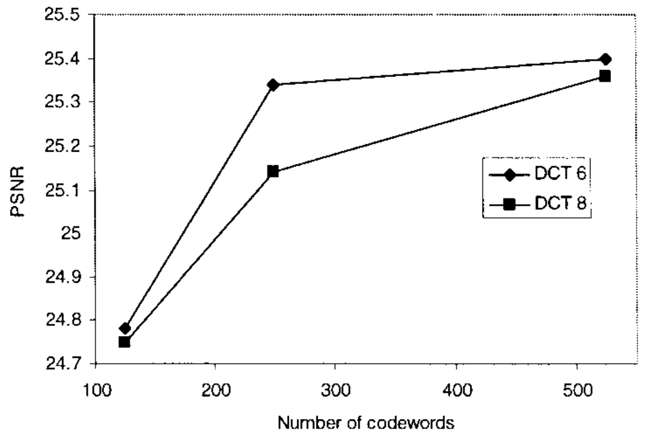

FIGURE 10 Peak signal-to-noise ratio evolution with the number of codewords.

Two remarks can be made:

- the compression rate does not vary too much with the cut frequency (Fig. 9): the difference is about 5% between DCT 6 and DCT 8;
- the signal to noise ratio (PSNR) does not vary too much with the cut frequency (Fig. 10). The difference between DCT 6 and DCT 8 is about 1%.

The fact that we obtain a better PSNR for the DCT 6 can be explained by the fact that the space quantized with DCT 6 has a lower dimension (dimension six); by this way, with a codebook of given size, the space can be better quantized than with the DCT 8 one (dimension equal to eight), with a codebook of fixed given size. Nevertheless, the image quality of the DCT 6 image, before the compression step, is lower (1.8 dB for Lena in Table I) than the DCT 8 one.

Differences between DCT 6 and DCT 8 in Fig. 9 are not significant, but the higher compression rate for DCT 8 could be explained by a better "organization" of the Kohonen map, because the input vectors (in dimension 8 instead of 6) better represent the "correlated" image vectors.

Figure 11 illustrates the compression rate (DCT 8) for both differential schemes, the first one using the previous block to compute the differences (Diffpred), the second one choosing the optimal direction to select the "previous" block (Dirmin).

Figures 12 and 13 respectively show the compressed Lena image and the difference image (with regards to the original one) without and with DCT (DCT 8); the visual quality remains similar while the PSNR is slightly decreased with the DCT (difference of 0.84 dB), but the compression rate is much increased: 14.22–25.22. Due to the use of Kohonen's self-organization property, the compression rate is increased to about 80%. It is an important result showing the effectiveness of our proposed compression scheme.

3.2. Lossless Compression

In these simulations, we used 8×8 blocks to benefit from a possible higher compression rate.

In Fig. 14, we can see that the DCT increases the compression rate of about 110%. In fact, if we analyze the simulations, we can see that the number of 0s transmitted to the entropy coder is more than six times higher in the case of the DCT. The UVLC takes full benefit from this and effectively increases the compression rate, which is now around 2.7 without any loss.

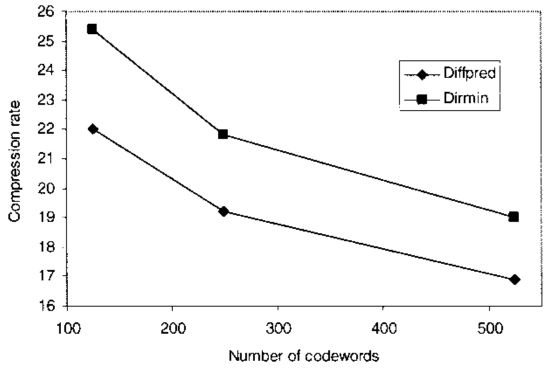

FIGURE 11 Comparison of the two proposed differential schemes.

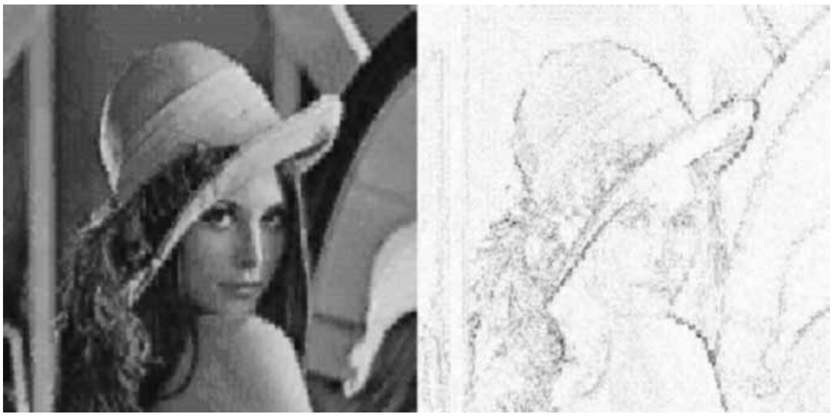

FIGURE 12 Lena image compressed without DCT, compression rate = 14.22, PSNR = 25.6 dB.

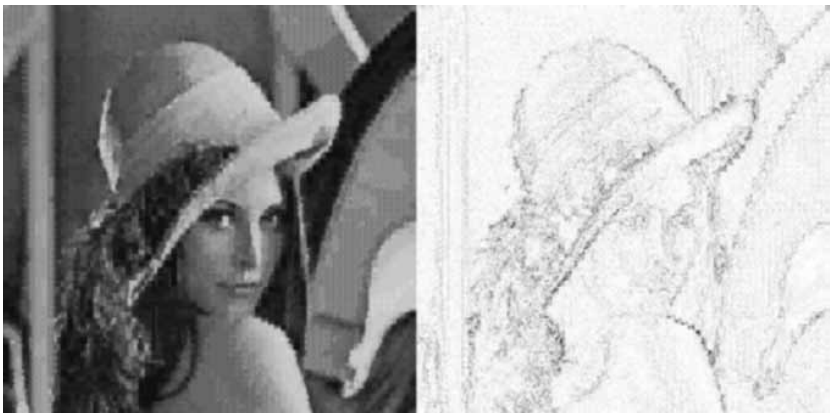

FIGURE 13 Lena image compressed with DCT, compression rate = 25.22, PSNR = 24.76 dB.

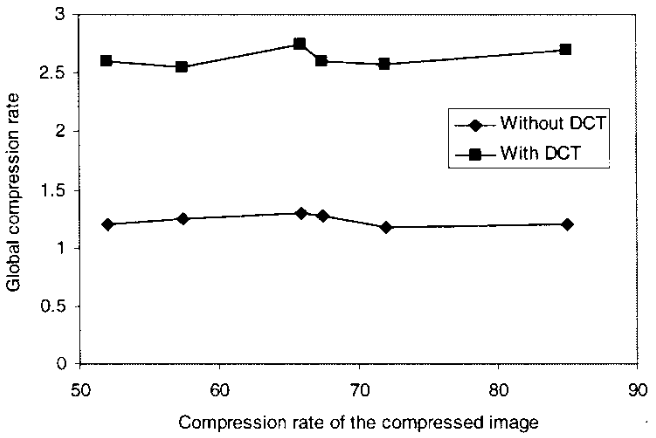

FIGURE 14 Compression rate with and without DCT for lossless compression.

3.3. Comparison with JPEG

To give an idea of the performances of the proposed coder–decoder using Kohonen maps, we compare our results with those obtained with the JPEG standard [17].

The JPEG's quantization matrix which was used is defined by:

$$P(u,v) = \mathrm{int}\left(Q(u,v)\frac{50}{quality} + 0.5\right) \tag{6}$$

where *quality* is a user-specified coefficient which allows to vary the compression rate. The $Q(u,v)$ matrix is defined by:

$$Q(u,v) = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix} \tag{7}$$

Figure 15 compares the compression rates obtained by our proposed compression scheme (Kohonen on the figure) and by the JPEG standard. We can see that our approach gives a higher PSNR as soon as the compression rate is higher than about 25. This simulation was carried out on the Lena image. From a visual point-of-view, we can compare the Lena image (and the difference image) compressed by our method and by the JPEG algorithm, respectively in Figs. 16 and 17, for a compression rate of about 25. The PSNR is slightly higher for JPEG, but the visual quality of the image is slightly better for our compression scheme. Figures 18 and 19 respectively show the Lena image compression with our method and with JPEG, for a compression rate of about 38. In this case, we can see that the PSNR is higher for our method, and claim with no doubt that the visual quality is much increased.

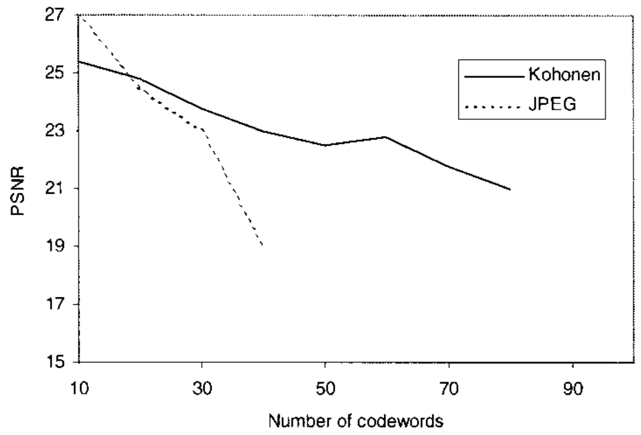

FIGURE 15 Comparison of PSNR for the proposed lossy compression scheme and the JPEG algorithm.

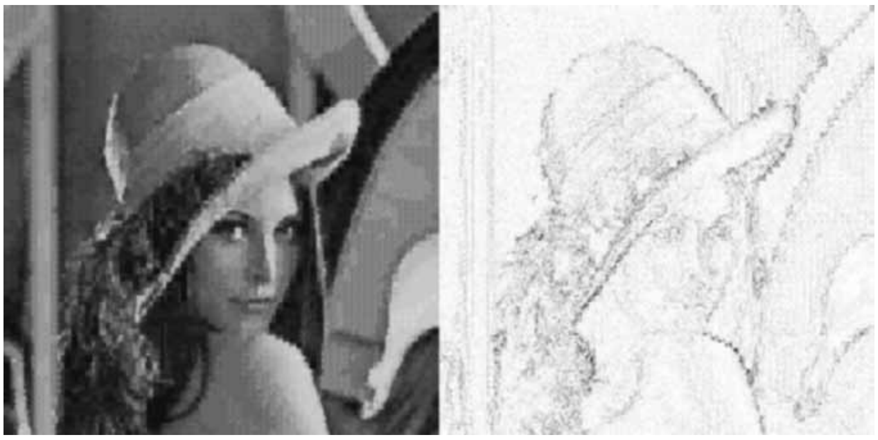
FIGURE 16 Lena image compressed by the proposed method, compression rate = 25.22, PSNR = 24.7 dB.

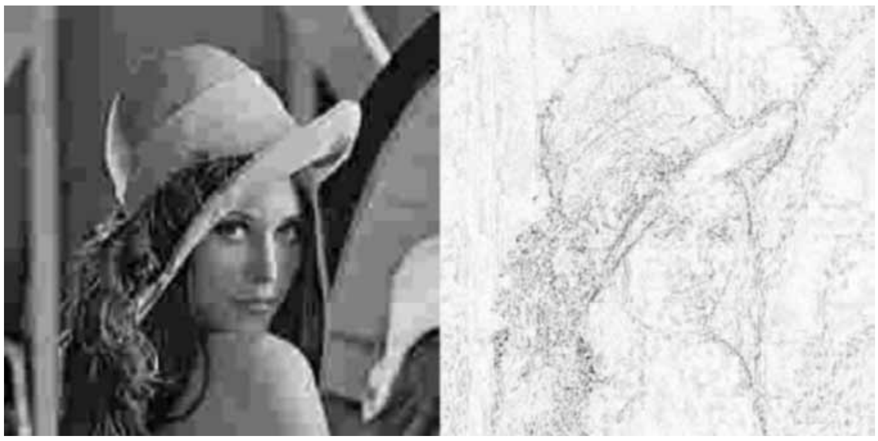
FIGURE 17 Lena image compressed by the JPEG algorithm, compression rate = 25.04, PSNR = 25.3 dB.

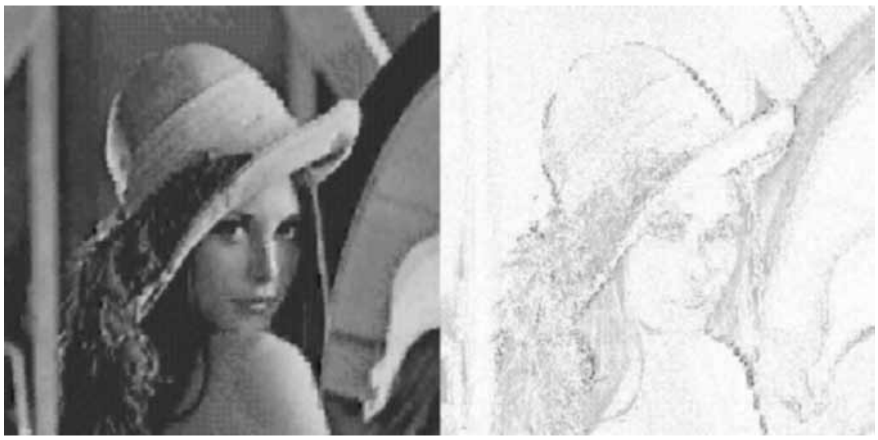
FIGURE 18 Lena image compressed by the proposed method, compression rate = 38, PSNR = 24 dB.

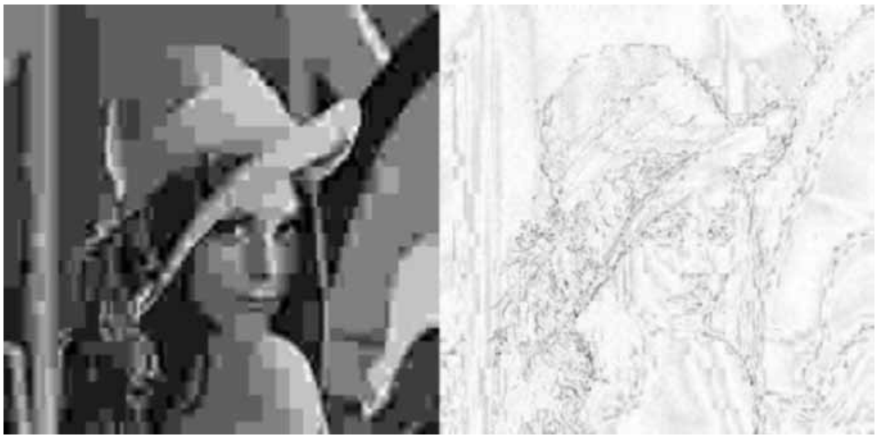

FIGURE 19 Lena image compressed by the JPEG algorithm, compression rate = 38.55, PSNR = 22.46 dB.

4. CONCLUSION

We proposed a new compression scheme based on the use of the organization property of Kohonen maps. It is based on the fact that consecutive blocks in an image are often similar, and thus coded by similar codewords with a vector quantization algorithm. The Kohonen map organization property makes the indexes of the coded vectors similar too, and, using an entropy coder, this property can be used to increase in a significant way the compression ratio, for a given image quality (in a lossy compression scheme). The same method can also be used in a lossless compression scheme. Comparisons with JPEG also show that the quality of a compressed image is better with our proposed scheme, for compression ratios greater than about 25.

Acknowledgment

Michel Verleysen is a research associate of the Belgian F.N.R.S. (National Fund for Scientific Research).

References

[1] C. Amerijckx, M. Verleysen, P. Thissen and J.-D. Legat (1998). Image compression by self-organized Kohonen map. *IEEE Transactions on Neural Networks*, **9**(3), 503–507.
[2] N. Ahmed, T. Natarajan and K.R. Rao (1974). Discrete cosine transform. *IEEE Trans. Com.*, C-**23**, 90–93.
[3] M.J. Narasimha and A.M. Peterson (1978). On the computation of discrete cosine transform. *IEEE Trans. Comm.*, COM-**26**, 934–936.
[4] R.M. Gray (1984). Vector qunatization. *IEEE ASSP Mag.*, 9–31.
[5] A. Gersho and R.M. Gray (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, London.
[6] T. Kohonen (1989). *Self-organization and Associate Memory*, 3rd Edn. Springer-Verlag, Berlin.
[7] T. Kohonen (1990). The self-organizing map. *Proc. of the IEEE*, **78**(9), 1464–1480.
[8] Y. Linde, A. Buzo and R.M. Gray (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, **28**, 84–95.
[9] N. Nasrabadi and Y. Feng (1988). Vector quantization of images based upon the Kohonen self-organizing feature maps. *IEEE International Conference on Neural Networks*, Vol. 1, pp. 101–108. SOS Printing, San Diego.
[10] E. de Bodt, M. Cottrell and M. Verleysen (1999). Using the Kohonen algorithm for quick initialization of simple competitive learning algorithm. *ESANN'99* (*European Symposium on Artificial Neural Networks*), pp. 19–26. Bruges, Belgium.

[11] E.A. Riskin, L.E. Atlas, and S.R. Lay (1991). Ordered neural maps and their applications to data compression. *Neural Networks for Signal Processing'91*, *IEEE Workshop*, pp. 543–551.

[12] S. Carroto, G.L. Sicuranza and L. Manzo (1993). Application of ordered codebooks to image coding. *Neutral Network for Signal Processing'93*, *IEEE Workshop*, pp. 291–300.

[13] B. Macq (1990). A universal entropy coder for transform on hybrid coding. *Picture Coding Symposium*, pp. 12.1.1–12.1.2. Boston.

[14] P. Delogne and B. Macq (1991). Universal variable length coder for an integrated approach to image coding. *Ann Telecomm.*, **46**(7–8), 452–459.

[15] A.C. Izquierdo, J.C. Sueiro and J.A. Méndez (1991). Self-organizing feature maps and their application to digital coding of information. In: A. Prieto (Ed.), *IWANN'91 Artificial Neural Networks*, Springer-Verlag Lecture Notes in Computer Sciences, pp. 278–283. Granada, Spain.

[16] G. Burel (1993). A new approach for neural networks: the scalar distributed representation. *Traitement du Signal*, **10**(1), 41–51.

[17] G.K. Wallace, (1991). The JPEG still picture compression standard. *Commun. ACM*, **34**(4), 32–34.