Neural Video Compression Algorithm

Michał Knop and Piotr Dobosz

Institute of Computational Intelligence, Czestochowa University of Technology,
Armii Krajowej 36, 42-200 Czestochowa, Poland
{michal.knop,piotr.dobosz}@iisi.pcz.pl

Abstract. In this paper, we present and discuss experimental results of a new concept of algorithm for video compression. It is named Predictive Vector Quantization (PVQ) and incorporates competitive neural network quantizer and neural network predictor. It is important for the image compression based on this approach to correctly detect scene changes in order to improve performance of the algorithm. We describe the scene detection algorithm based on the image entropy and discuss its effectiveness.

1 Introduction

Multimedia data transmission is widely spread nowadays. Most of the applications require effective data compression in order to decrese the required bandwidth or storage space. Various techniques of coding of the data achieve this goal by reducing data redundancy. In most of the algorithms and codecs a spatial compensation of images as well as movement compensation in time is used. Video compression codecs can be found in such applications as:

1. various video services over the satellite, cable, and land based transmission channels (e.g., using H.222.0 / MPEG-2 systems);
2. by wire and wireless real-time video conference services (e.g., using H.32x or Session Initiation Protocol (SIP) [1]);
3. Internet or local area network (LAN) video streaming [2];
4. storage formats (e.g., digital versatile disk (DVD), digital camcorders, and personal video recorders) [3].

Currently, many image and video compression standards are used. The most popular are JPEG and MPEG. They differ in the level of compression as well as application. JPEG and JPEG2000 standards are used for image compression with an adjustable compression rate. There is a whole family of international compression standards of audiovisual data combined in the MPEG standard, which is described in more details in literature [4]. The best known members are MPEG-1, MPEG-2, and MPEG-4. We used a PVQ (Predictive Vector Quantization) algorithm in our work to compress a video sequence. It combines a VQ (Vector Quantization) [5,6] and DPCM (Differential Pulse Code Modulation). More information on the techniques can be found in sources [7,8,9]. To detect a scene change we used method based on the entropy. Then we can change necessary parameters of the predictor and the codebook.

R.S. Choraś (ed.), *Image Processing & Communications Challenges 6*,
Advances in Intelligent Systems and Computing 313, DOI: 10.1007/978-3-319-10662-5_8

2 Video Compression Algorithm

The design of the compression algorithm described here is based on the existing algorithm described in [7,8,9]. We chose this algorithm because thanks to neural network it exhibits better adjustment parameters to a frame and gives better image quality after compression. The extension includes a scene change detection algorithm, which is based on the entropy of frames. The diagram below Fig. 1. shows the proposed algorithm.

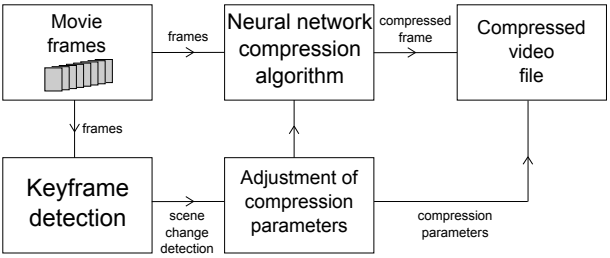

Fig. 1. Video compression algorithm

2.1 Neuronal Image Compression

Fig. 1 [9] shows the architecture of the PVQ. It combines the Huffman coding with a vector extension of the scalar differential pulse code modulation scheme [7,8]. The block diagram of the PVQ algorithm consists of the encoder and decoder, each containing: an identical neural-predictor, a codebook, a neural vector quantizer, the Huffman coder.

The successive input vectors $\mathbf{V}(t)$ are introduced to the encoder. The difference $\mathbf{E}(t) = [e_1(t), e_2(t), ..., e_q(t)]^T$ given by the equation

$$\mathbf{E}(t) = \mathbf{V}(t) - \overline{\mathbf{V}}(t) \tag{1}$$

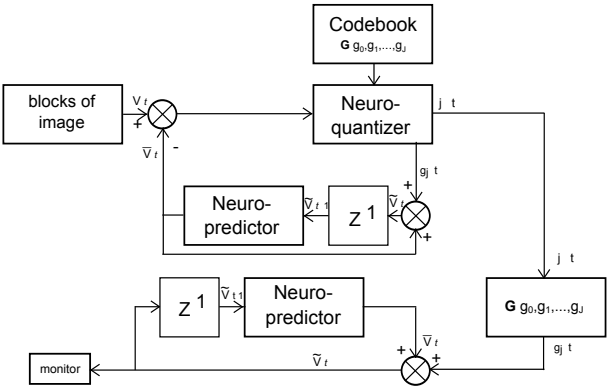

Fig. 2. The architecture of the image compression algorithm

is formed, where: $\overline{\mathbf{V}}(t) = [\overline{v}_1(t), \overline{v}_2(t), ..., \overline{v}_q(t)]^T$ is the predictor of $\mathbf{V}(t)$. Statistics shows that fewer quantization bits are required by the difference $\mathbf{E}(t)$ than the original subimage $\mathbf{V}(t)$. The next step is vector quantization of $\mathbf{E}(t)$ using the set of reproduction vectors $\mathbf{G} = [\mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_J]$ (codebook), where $\mathbf{g}_j = [g_{1j}, g_{2j}, ..., g_{qj}]^T$ (codewords). For every q-dimensional difference vector $\mathbf{E}(t)$, the distortion (usually the mean square error) between $\mathbf{E}(t)$ and every codeword $\mathbf{g}_j$, $j = 0, 1, ..., J-1$ is computed. The codeword $\mathbf{g}_{j^0}(t)$ is selected as the representation vector for $\mathbf{E}(t)$ if

$$d_{j^0} = \min_{0 \le j \le J} d_j, \tag{2}$$

a measure d in expression (2)we can take e.g. the Euclidean distance. Observe that by adding the prediction vector $\overline{\mathbf{V}}(t)$ to the quantized difference vector $\mathbf{g}_{j^0}(t)$ we get the reconstructed approximation $\widetilde{\mathbf{V}}(t)$ of the original input vector $\mathbf{V}(t)$, i.e.

$$\widetilde{\mathbf{V}}(t) = \overline{\mathbf{V}}(t) + \mathbf{g}_{j^0}(t). \tag{3}$$

The prediction vector $\overline{\mathbf{V}}(t)$ of the input vector $\mathbf{V}(t)$ is made from past observation of reconstructed vector $\widetilde{\mathbf{V}}(t-1)$. In our algorithm, the predictor is specifically designed for this purpose as a nonlinear neural network. As the last stage, the set of the $j^0(t)$ is coded by the Huffman coder. The codebook of the Huffman coder is designed using a set of counters f_j which count how frequently given label $j^0(t)$ occurs after presentation of all vectors $\mathbf{V}(t)$. The appropriate codewords $h^0(t)$ from the Huffman codebook are broadcasted via the transmission channel to the decoder, where they are decoded and the reconstructed vector $\widetilde{\mathbf{V}}(t)$ is formed in the same manner as in the encoder (see formula (3)).

2.2 Scene Detection

Among many other approaches [10,11,12], the methods based on the entropy are worth to take into account. We were used the phenomenon of the entropy to determine the complexity of information stored in each film frames. The general formula for the entropy on the greyscale video materials looks like:

$$E_j = -\sum_{i=0}^{N} p_i log(p_i), \tag{4}$$

where E_j is a characteristic value calculated from current frame, p_i is the next pixel from this frame, and N is a number of whole pixels in current image. Since we have been used *HSV* color gamut, we must calculate the value of the entropy for each coefficient in palette. The individual components of the *HSV* model aren't equal. The hue *(H)* has the biggest importance in the process of determining the coefficient. Even the slightest change of its value should lead to detecting scene changes. A bit less importance has the color saturation *(S)*, which even larger changes are for us fully acceptable. The value of the color *(V)*

has for us the lowest importance. Hence the final value of entropy for each frames in the *HSV* color space will be calculated from the formula:

$$F_n(E) = a \cdot E_j(H) + b \cdot E_j(S) + c \cdot E_j(V) \tag{5}$$

Parameters of a, b and c corresponds to the weighting factors for the entropy component of *HSV*. For our experiment, the values are successively set to values 0.9, 0.3 and 0.1 [13]. To detect new keyframe we compare current keyframe with subsequent frames. The following formula describes used compare operation:

$$E_{j,k} = \frac{|E_j - E_k|}{E_k}, \tag{6}$$

where j is a number of our current frame and k is a current key frame. The first key frame is always the first image taken at start of the film. The next key frames are being calculated based on the comparison result (see formula 6). The threshold value for determining a keyframe was selected experimentally and is set to be -0.2. If the entropy value is lower than the assumed threshold, the algorithm determines a new key frame.

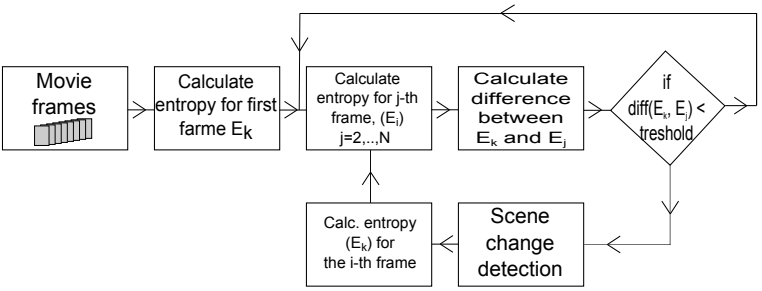

Fig. 3. Scene change detection algorithm

3 Experimental Result

The effectiveness of the algorithm was tested on a set of uncompressed frame captured by a digital USB camera of a 640×480 resolution with 256 levels of grey. Four tests were conducted. In the first and second test, we used the frames within a single scene (Fig. 4). In the first test, the frames were compressed creating separate codebook and the predictor for each frames (Fig. 5). For the second test we used the same codebook and the predictor for all frames(Fig. 6).

A transit frames between scenes were chosen for the third and fourth tests based on the scene change detection algorithm (Fig. 7). In this algorithm each frame is compare with the keyframe. When the new scene is detected the algorithm marks a new keyframe (Fig. 8). Thanks to the training of predictor and codebook for the keyframe, algorithm adapts better compression parameters to the set of frames and improves the image quality after decompression (Fig. 9).

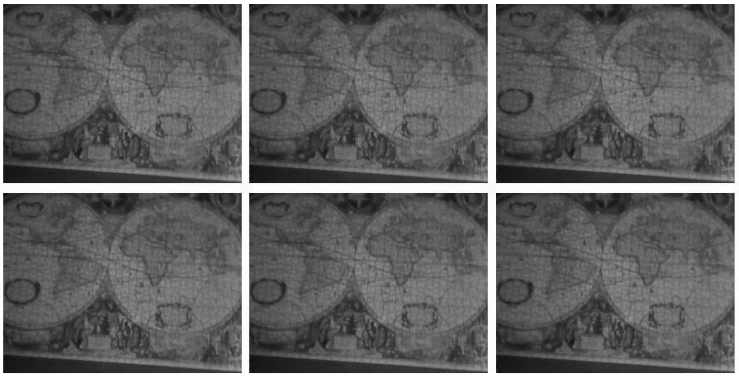

Fig. 4. a)original sequence b)compressed sequence test 1 c)compressed sequence test 2

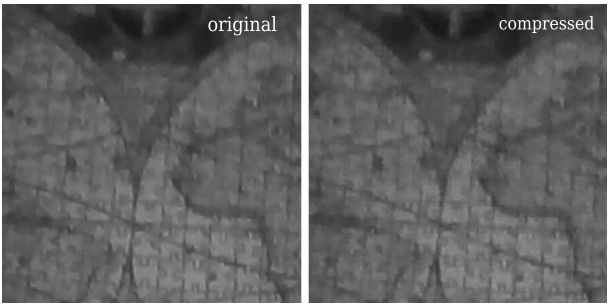

Fig. 5. Difference between frames in test 1

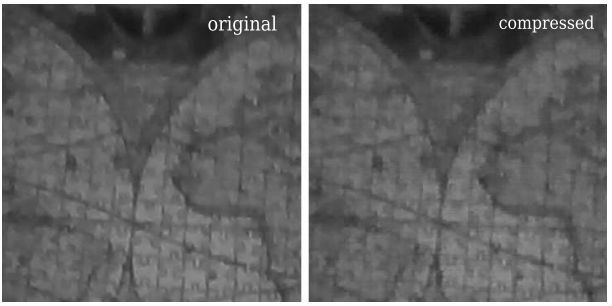

Fig. 6. Difference between frames in test 2

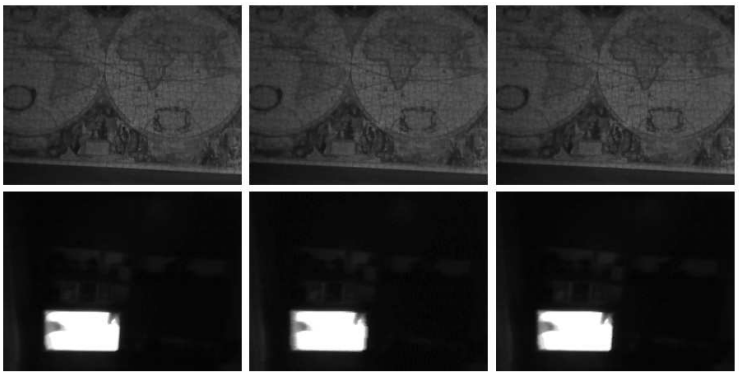

Fig. 7. a)original sequence b)compressed sequence test 3 c)compressed sequence test 4

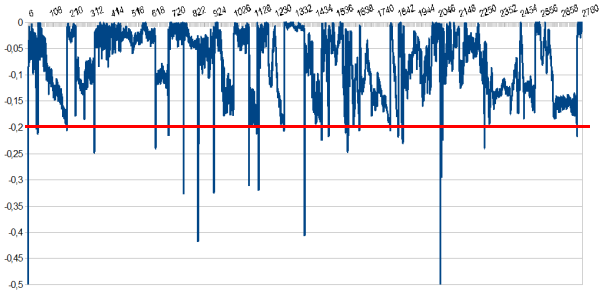

Fig. 8. Scene change detection

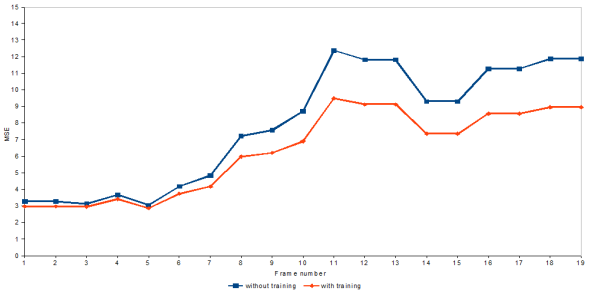

Fig. 9. MSE change

In test 3 the same codebook and predictor were used before and after the scene transit. As the results show, this approach is insufficient in case of a major scene change (Fig. 10). For the fourth test, the scene transition was detected and separate codebooks and predictors were created for frames before and after the scene transition (Fig. 11).

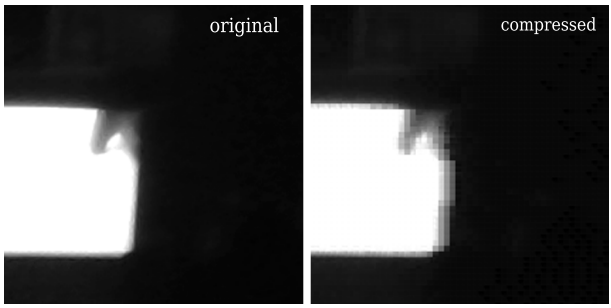

Fig. 10. Difference between frames in test 3

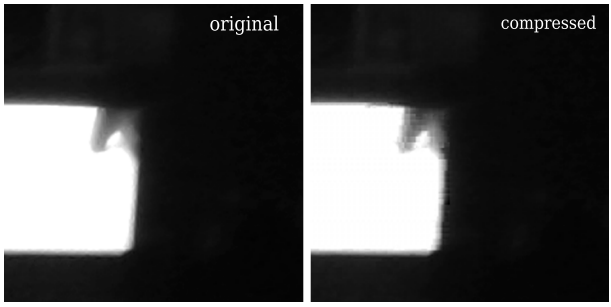

Fig. 11. Difference between frames in test 4

4 Conclusions

The tests show that the scene change detection algorithm is especially useful for the presented compression algorithm. It is apparent that without the scene detection video sequence compressed by our algorithm would exhibit a poor quality of frames after the scene transition. On the other hand, the amount of data resulting from including the compression parameters for every frame would greatly impact the output file's size.

References

1. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol. Internet Eng. Task Force (IETF). Request for Comments (RFC), 3261 (2002)

2. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A transport protocol for real-time applications. Internet Eng. Task Force (IETF). Request for Comments (RFC), 1889 (1996)
3. Sullivan, G.J., Wiegand, T.: Video Compression—From Concepts to the H.264/AVC Standard. Proceedings of the IEEE 93, 18–31 (2005)
4. Clarke, R.J.: Digital compression of still images and video. Academic Press, London (1995)
5. Gray, R.: Vector quantization. IEEE ASSP Magazine 1, 4–29 (1984)
6. Gersho, A., Gray, R.M.: Vector quantization and signal compression. Kluwer Academic Publishers (1992)
7. Rutkowski, L., Cierniak, R.: Image compression by competitive learning neural networks and predictive vector quantization. Applied Mathematics and Computer Science 6, 706–711 (1996)
8. Cierniak, R., Rutkowski, L.: On image compression by competitive neural networks and optimal linear predictors. Signal Processing: Image Communication 15, 559–565 (2000)
9. Cierniak, R.: An Image Compression Algorithm Based on Neural Networks. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 706–711. Springer, Heidelberg (2004)
10. Chen, L., Feris, R., Turk, M.: Efficient partial shape matching using Smith-Waterman algorithm. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW, pp. 1–6 (2008)
11. Adhikari, P., Gargote, N., Digge, J., Hogade, B.G.: Abrupt Scene Change Detection. World Academy of Science, Engineering and Technology. International Science Index 18, 687–692 (2008)
12. Seeling, P.: Scene Change Detection for Uncompressed Video. In: Technological Developments in Education and Automation, pp. 11–14 (2010)
13. Qu, Z., Lin, L., Gao, T., Wang, Y.: An Improved Keyframe Extraction Method Based on HSV Colour Space. Journal of Software 8, 1751–1758 (2013)