Hybrid PCA Algorithm for Image Compression

Stefan Stolevski

***Abstract* – In this paper a method based on Principal Component Analysis for image compression is proposed. The method effectively combines two algorithms in one hybrid algorithm. PCA is a technique which makes use of the correlation in data in order to achieve dimensionality reduction. The first algorithm uses the correlation between the three components a color image is composed of, while the second one uses the correlation between the blocks a gray image is composed of.**

***Keywords* – Compression ratio (CR), Image compression, Peak Signal to Noise Ratio (PSNR), Principal Component Analysis (PCA)**

I. INTRODUCTION

THE need for efficient data compression is growing, primarily due to the need for quick transfer and saving memory capacity. The dimensionality reduction problem is directly related to image compression. Principal component analysis (PCA) also known as Karhunen-Loeve expansion, is one of the classical dimensionality reduction methods used for feature extraction which has been widely used in variety of areas such as signal processing, pattern recognition and data mining. PCA has been widely applied in the area of image compression in various forms, i.e. as a standalone image compression technique as well as a pre-processing or post-processing step in combination with other techniques.

There are developed algorithms for image compression using the popular *DCT (Discrete Cosine Transform)* or wavelet transform as in [1], where the transformation matrix is known in advance and independent of the input image. These algorithms achieve suboptimal decorrelation, but they are relatively quick and simple. *PCA* is a linear transformation based on statistical methods. There are papers on image compression using two popular basic one-dimensional approaches, but they lack more precise evaluation and the algorithms are treated separately as in [2],[4] or a complex modification of one of them is proposed, as in [3]. A detailed analysis of two-dimensional *PCA* is shown in [4]. In this paper we evaluate the two *PCA* algorithms, and propose a hybrid algorithm obtained by combining them.

II. THE GENERAL PCA ALGORITHM

As already stated, the aim of *PCA* is data dimensionality reduction. Accordingly, a multidimensional data set with dimension N is given, i.e. a set composed of N rows. First, we find the mean of each row and subtract this value from the corresponding row. The mean subtracted is the average across each dimension. We call the resulting matrix the *adjusted matrix A*. We find its *covariance matrix (NxN)* and then calculate its eigenvalues and eigenvectors. These are rather important, as they tell us useful information about our data, i.e. they provide us with information about the patterns in the data. Next, the eigenvectors are sorted according to their significance (the absolute values of the corresponding eigenvalues) and a matrix F composed of eigenvectors is formed, where each vector is a column. If we want to achieve compression (and that is our primary goal), we discard a number of eigenvectors starting from the least important one. The number of vectors (k) we keep depends on the loss of information we can afford, regarding the particular application. To obtain the new, compressed data set, we transpose the eigenvector matrix and multiply it by the adjusted matrix:

$$A_k = F_k^T * A$$

Now the data is compressed and presented in a different coordinate system. To return to the original coordinate system, i.e. to obtain the original data (decompressed), we have (keeping in mind that F_k is an orthogonal matrix):

$$A = (F_k^T)^{-1} * A_k$$

If $k=N$ (no loss), then

$$A = F_k * A_k$$

$$A_{original} = F_k * A_k + M$$

else

$$A' = F_k * A_k$$

$$A'_{original} = F_k * A_k + M$$

where M is a matrix of the mean values (averages) of the input dimensions. Hence, we have input set $A_{original}$(dimension N), compressed set A_k(dimension k) and decompressed set $A'_{original}$(dimension N) where $A'_{original}=A_{original}$ if $k=N$. F_k, A_k and M are the only values we need to store, since they are sufficient in obtaining $A'_{original}$.

III. COLOR IMAGE COMPRESSION

Each color image consists of three components whether stored in *RGB* or *YCbCr* format. For this application, the input image is first converted into *YCbCr* format as in [6], because in this way additional decorrelation of the components (better compression) is done. Each component of an image represents one dimension in *PCA* (let us call it InterPCA) as in [2]. So $N = 3$, $q = 1,2,3$. Instead of doing PCA on the whole image, the image is divided into several sub images or blocks and *PCA* is done on each 3-component block individually. Hence, each block represents input data set in *PCA* and undergoes the transformations presented above. Since one dimensional *PCA* is used, we first need to transform each two-dimensional color component from the block into row by concatenating the rows one after the other so the input data

Stefan Stolevski is a graduate student at the Faculty of Electrical Engineering and Information Technologies, Skopje (phone: +389 71 227 626; e-mail: stefanstolevski@gmail.com).

set has the required format. At output, we get a block with reduced dimension A_q (composed of one or two color components depending on q), q eigenvectors (three values each, i.e $N=3$) and the averages (three values). This is all that is needed for image reconstruction (decompression). Let the blocks be *nxn* square blocks and let the image size be *MxN*. Then, the total number of blocks is MN/n^2 In order to calculate the CR, we need to divide the total data needed to represent the input image by the total data needed to represent the output of *InterPCA*. Thus, the CR is:

$$CR = \frac{3MN}{qMN + \frac{MN}{n^2}(3q+3)}$$

For $q = 3$, $CR < 1$ (expansion) which has no practical use and is not considered in the following text.

IV. Gray Image Compression

Similarly to the *InterPCA*, in *IntraPCA* we divide a gray image, which consists of only one component, into identically sized blocks. Now each block represents one dimension in *PCA,* as in [4]. As before, we first transform all the blocks into rows. Thus, when compressing, we discard all blocks (now rows) for which there are similar, i.e. we can approximately regenerate the image using the remaining (not discarded) blocks. At output, we get an image with fewer blocks (reduced dimension), as well as the eigenvectors and averages. Let the blocks be *nxn* square blocks and let the image size be *MxN*. Then the total number of blocks is MN/n^2. For decompression, we need the first k blocks of *PCA*, a total of k eigenvectors (MN/n^2 values each - as the input dimensionality), plus MN/n^2 values (averages) . So

$$CR = \frac{MN}{kn^2 + k\frac{MN}{n^2} + \frac{MN}{n^2}}$$

V. Hybrid Algorithm

By merging the two algorithms in a cascade, a hybrid algorithm for color image compression is proposed. First, we use the first *PCA* compression algorithm (*interPCA*) after which we get one or two components, depending on q (instead of three), together with all their eigenvectors and averages (Fig.1.). Then, each one of the components undergoes compression with the second algorithm (*intraPCA*), whereby the output again consists of image (images) with reduced dimension (fewer blocks) and their eigenvectors and averages. For decompression, the output from the hybrid algorithm is required. From the second algorithm we have A_k, i.e. qkn^2 values, eigenvectors $qk\frac{MN}{n^2}$ and averages $q\frac{MN}{n^2}$. Using these data, after decompression, we get q image components. For a complete decompression, the eigenvectors and averages from the first algorithm ($3q\frac{MN}{n^2}$ and $3\frac{MN}{n^2}$) are required . From the discussion above, the total CR is:

$$CR = \frac{3MN}{q\left(kn^2 + k\frac{MN}{n^2} + \frac{MN}{n^2}\right) + \frac{MN}{n^2}(3q+3)}$$

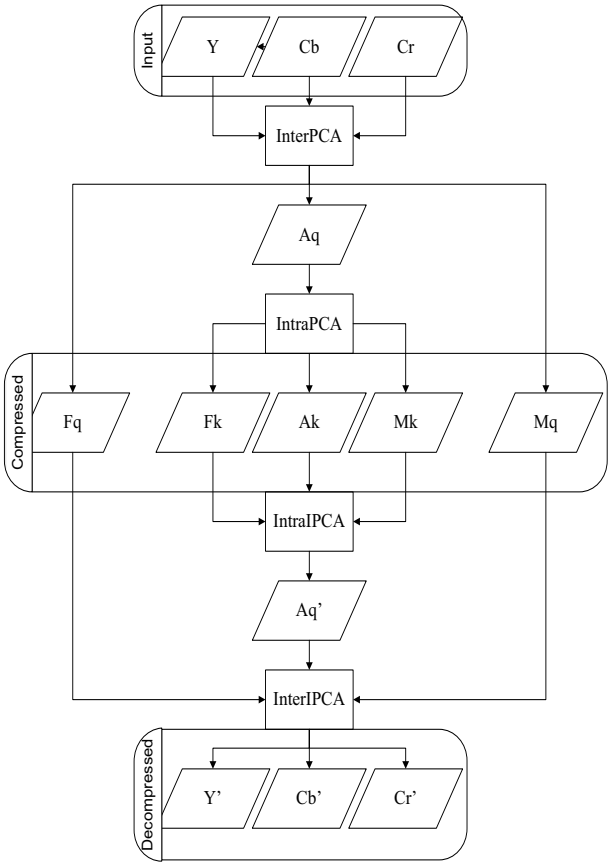

Fig. 1. Block diagram of the proposed algorithm

VI. Experimental Results

For the implementation and evaluation of the algorithms we developed a MATLAB code and performed the testing on a standard color test image *Lena 512x512* and another test image *Hdr 2048x3072*. We analyze the results obtained with the first, second and finally, the proposed hybrid algorithm. All images and tables from the experiments are given. Standard measures for image compression like CR and PSNR were used, defined in [1]. For the first algorithm, in Table 1, we can see the values of *CR* and *PSNR* for different values of n and $q = 1$ and in Table 2, for $q = 2$. Subjective quality does not always correspond to the value of PSNR. For $q = 1$, for PSNR of *30-40 dB* one might expect relatively high quality image but because of the nature of the algorithm, blocking effect in coloration appears (Fig. 2). Perceptually good results are obtained for $n < 16$. For $q = 2$ (Table 2), the compressed image is visually identical to the original for $n < 256$. However, this case is impractical due to the relatively small CR.

For the second algorithm, testing is carried out only for n = *16,32,64* because for larger values of n the blocking effect is obvious, whereas for smaller values the CR is impractically small and the computing time large. From tables 3,4 and 5 and Fig. 3,4 and 5 it can be concluded that quality compression is obtained for small values of n (on the expense of computing time). For example, the value of *CR* and *PSNR* for $n = 16$, $k = 32$ is approximately equal to that for $n = 32$, $k = 32$, and the visual difference is not negligible. Furthermore, for $n = 64$, $k = 24$

TABLE 1

q=1		
n	CR	PSNR
512	3.00	35.50
256	3.00	36.22
128	3.00	37.54
64	3.00	39.86
32	2.98	41.69

TABLE 2

q=2		
n	CR	PSNR
512	1.50	46.41
256	1.50	47.00
128	1.50	49.00
64	1.50	50.21
32	1.49	50.95

TABLE 3

n=16		
k	CR	PSNR
32	6.24	35.98
64	3.16	38.58
96	2.12	41.35

TABLE 4

n=32		
k	CR	PSNR
32	6.36	34.08
64	3.19	36.61
96	2.13	39.61

TABLE 5

n=64		
k	CR	PSNR
8	7.86	30.73
16	3.93	31.88
24	2.62	33.25

TABLE 6

n=64		
k	CR	PSNR
128	8.71	43.72
256	4.36	48.03
384	2.91	51.09

TABLE 7

n=16, q=1		
k	CR	PSNR
16	28.44	33.98
32	16.34	34.76
64	8.83	35.99

TABLE 8

n=16, q=2		
k	CR	PSNR
16	15.00	34.26
32	8.44	35.33
64	4.49	37.40

TABLE 9

n=64, q=1		
k	CR	PSNR
64	50.85	38.07
128	25.80	39.27
192	17.28	39.92

the value of *CR* is relatively small and the quality quite poor, which confirms the conclusion.

This algorithm was also tested on a high resolution test image (*Hdr*) (Fig. 6). From Table 6 and Fig. 6 and 7 it is obvious that here the compression is significantly more efficient than the one with the low resolution image (*Lena*), even though the image is more complex in texture. It is so, because now the *PCA*, which is statistically based, has more data to work with. Hence, when obtaining the principal components (the most important blocks) a more precise statistical model for the image is used. The improvement in quality is not only visual, i.e. only due to the higher resolution (a phenomenon that occurs in *JPEG*). For example, here at *CR = 8.71 PSNR = 43.72*, while for the lower resolution image (*Lena*), *PSNR = 41.35* for *CR = 2.12* (Table 3).

The hybrid algorithm was also implemented and tested in MATLAB using the popular *Lena* image. For *n>16*, the blocking effects in color are significant and therefore only $n \leq 16$ are taken into consideration. Comparing the first row of Table 7 (*n = 16, q = 1, k = 16*) with the first row of Table 8 (*n = 16, q = 2, k = 16*), one can notice that the compression in Table 7 is more efficient, so as before, the best choice is *q = 1*. Fig. 8 and 9 are given as an example of a good compression. In order to confirm the conclusion that compression is more effective for images with higher resolution, the algorithm is tested on such an image. In this case (Table 9, Fig. 10), at *PSNR = 38, CR = 50.8*.

Fig. 2. q=1, n=64

Fig. 3. n=16, k=32

Fig. 4. n=32, k=32

Fig. 5. n=64, k=24

As a suggestion for future work, blocking effects which occur in color (Fig. 11) can be eliminated by choosing different values for *n* in *interPCA* and *intraPCA,* i.e. smaller values for *interPCA*. Another solution would be a digital filter (taking into consideration that the inside of each block is of high quality). Also, different quantization schemes and coding could be used for efficiency improvement.

Fig. 6. Original 2048 x 3072

Fig. 7. n=64, k=128

Fig. 8. n=16, k=16, q=1

Fig. 9. n=16, k=32, q=1

Fig. 10. n=64, k=64, q=1

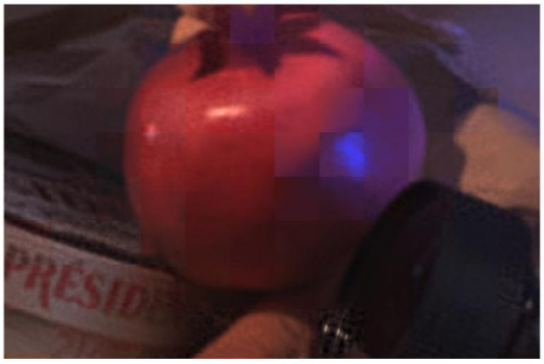

Fig. 11. Blocking effect

VII. CONCLUSION

Using the proposed Hybrid algorithm we can achieve CR of over 25 with negligible loss of quality, especially with images with high resolution. For comparison, with *JPEG* (where entropy coding is included), CRs of up to 10-15 with negligible loss are achieved**.** In [3], the authors mixed the color and texture information into a single vector and achieved PSNR values of about 40dB for CR of about 20, whereas here, PSNR =38 for CR=50 (for high resolution images). The main drawback is its complexity, which is too high for real-time applications.

REFERENCES

[1] Amhamed Saffor, Abdul Rahman Ramli, Kwan-Hoong Ng, "A Comparative Study of Image Compression Between JPEG and Wavelet", Malaysian Journal of Computer Science, Vol. 14 No. 1, June 2001, pp. 39-45

[2] M. Mudrova, A. Prochazka,"Principal Component Analysis in Image Processing", Institute of Chemical Technology, Prague Department of Computing and Control Engineering, unpublished

[3] D. Carevic, T. Caelli, "Region–based coding of color images using karhunen–loeve transform", Graphical Models and Image Processing 59(1) (1997) 27–38

[4] Ashutosh Dwivedi, Arvind Tolambiya, Prabhanjan Kandula, N Subhash Chandra Bose, Ashiwani Kumar, Prem K Kalra," Color Image Compression Using 2-Dimensional Principal Component Analysis (2DPCA)", Department of Electrical Engineering, Indian Institute of Technology Kanpur, Uttar Pradesh, India-208016, Proc. of ASID '06, 8-12 Oct, New Delhi

[5] Arash Abadpour, Shohreh Kasaei,"Color PCA Eigenimages and their Application to Compression and Watermarking", submitted to Image & Vision Computing 21 August 2007