PERGAMON

Pattern Recognition 33 (2000) 1555–1560

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Color image compression using PCA and backpropagation learning

Clifford Clausen, Harry Wechsler*

Department of Computer Science, George Mason University, Fairfax, VA 22030, USA

Received 6 April 1999; accepted 17 May 1999

Abstract

The RGB components of a color image contain redundant information that can be reduced using a new hybrid neural-network model based upon Sanger's algorithm for representing an image in terms of principal components and a backpropagation algorithm for restoring the original representation. The PCA method produces a black and white image with the same number of pixels as the original color image, but with each pixel represented by a scalar value instead of a three-dimensional vector of RGB components. Experimental results show that as our hybrid learning method adapts to local (spatial) image characteristics it outperforms the YIQ and YUV standard compression methods. Our experiments also show that it is feasible to apply training results from one image to previously unseen images. © 2000 Published by Elsevier Science Ltd.

Keywords: Principal component analysis (PCA); Color image compression; Backpropagation (BP) learning

1. Introduction

Even though the spectral content of illumination is infinite dimensional, three dimensions are enough to adequately represent color. Judd et al. [1] use principal component analysis of daylight illumination to show that 99% of the variance can be accounted for with only three principal components. Furthermore, 85% of variance can be represented with only two color channels. Today, color images are rendered on computer monitors using only three primary colors, usually red, green and blue (RGB). Therefore, a straightforward way to compress a color image is to compress each of the red, green and blue gray-scale images that compose the image.

A common alternative to the RGB representation is the YIQ representation, which is the standard used for television transmission. An RGB represented color image can be converted to YIQ coordinates as follows [2, pp. 45–46]:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ 0.60 & -0.27 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

Here Y is the luminance or brightness, I is the hue, and Q is the saturation or depth of color. Luminance refers to color intensity. Hue is the dominant color such as orange, red or yellow and saturation is the amount of white light mixed with a hue. With respect to a television set, Y represents the black and white image and is what is displayed by a black and white television set. I and Q correspond to the two color adjustment knobs found on a color television set and contain the additional information used to produce color images. For this reason, I and Q together are sometimes referred to as chrominance values. An RGB representation can be obtained from a YIQ representation by inverting the above transformation as

* Corresponding author.

0031-3203/00/$20.00
PII: S0031-3203(99)00126-0

follows:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.00 & 0.96 & 0.62 \\ 1.00 & -0.27 & -0.65 \\ 1.00 & -1.10 & 1.70 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}.$$

Television transmission uses less bandwidth for the hue and saturation components than it does for luminance. Similarly, when compressing color images, the two chrominance values, I and Q, can be decimated by between one-quarter and one-half without perceptible image degradation [2]. It is also possible to decimate Y as results shown in Section 3 below. Through decimation, the number of bits required to represent the original color image can be reduced by as much as $\frac{1}{3}$ or more before gray-scale encoding begins. Encoding a color image then consists of converting an RGB image to YIQ coordinates, decimating Y, I and Q and then applying a gray-scale compression algorithm such as JPEG [3, pp. 347–408] to each of the Y, I, and Q coordinates. To reconstruct a color image, decode each of Y, I, and Q and then interpolate the Y, I and Q to their original size. Finally, use the inverse transformation above to obtain RGB coordinates for the image.

Another color transformation is the YUV code used by MPEG 1. Again Y is brightness or luminance and U and V are chrominance values. The luminance Y is defined as before, but the color information is stored differently. The YUV transformation is defined as follows:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ -0.15 & -0.29 & 0.44 \\ 0.61 & -0.52 & -0.1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

As before, we can convert a YUV representation to an RGB representation using the inverse transformation as follows:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.01 & 0.007 & 1.14 \\ 0.994 & -0.381 & -0.583 \\ 1.00 & 2.02 & 0.006 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}.$$

We obtain compression from YUV coordinates by decimating Y, U, and V. As pointed out in Ref. [4], there are other color coordinate systems that can be used to represent color such as the Munsel color order system [5] and CIELUV [6]. These color coordinate systems do not take advantage of the spatial relationships between colors in an image. The colors of adjacent pixels in a natural image are highly correlated and, because of this spatial correlation, further compression is possible. In the next section, a new method for encoding color images using a learning method based upon principal component analysis which takes advantage of spatial correlation of color is presented.

2. Learning method

The general approach is to pre-process a color image by first reducing its dimension using learning based upon principal component analysis (PCA) and then compressing the reduced image. We reduce the dimension of an image by first partitioning it into 2×2 blocks (that is 4 pixels per block). The color image is assumed to be represented in RGB format and hence each pixel is represented by a three-dimensional vector, (r, g, b), where each vector component is represented by one byte. Therefore, each 2×2 block can be represented as a (3 components per pixel) $\times$ (4 pixels) = 12-dimensional vector, where each vector component is one byte. We consider an image to be a collection of 12-dimensional vectors and our objective is to find a partial basis to represent this 12-dimensional vector space. We want the partial basis to minimize the sum of square errors over the entire image. We will use a partial basis consisting of four orthogonal vectors as this works well experimentally. The space represented by this partial basis will be called the reduced image space and the original image transformed to the reduced image space will be called the reduced image.

In matrix notation, we are seeking a matrix $A_{4\times 12}$ such that $Ax = y$ where x is a 12-dimensional vector in the original image with the mean, μ_{12} removed, and y is a 4-dimensional vector in the reduced image. It is the reduced image that we compress using a gray-scale compression method. Let $B_{12\times 4} = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$ be the pseudo inverse of A so that $\hat{x} = By$. Then the B matrix should be such that the square error defined as $J = \sum_{i=0}^{N-1} (By_i - x_i)^{\mathrm{T}}(By_i - x_i)$ is minimized where there are N 2×2 blocks in the image, x_i corresponds to the ith 2×2 pixel block in the image and By_i, is the estimate $\hat{x}_i$ of x_i. We will use a neural network learning algorithm based on PCA to determine A and use a feed forward neural network trained with back propagation to determine B. We connect these two networks together to form a hybrid neural network as depicted in Fig. 1.

The Sanger Network [7] in Fig. 1 creates a reduced dimension image and the feed forward network (FFN) produces an estimate of the original color image from the reduced image. The Sanger network is trained using Sanger's unsupervised learning algorithm. The feed forward network is trained using a supervised learning, back propagation algorithm [8]. The hybrid network is trained using the 12-dimensional vectors x as input vectors to learn the A matrix using Sanger's rule. The A

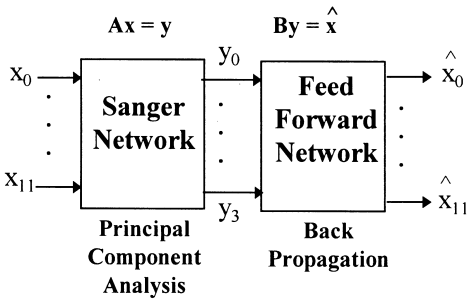

Fig. 1. Color reduction network.

matrix is used to calculate four-dimensional vectors, y, for each input vector x. The y vectors are then used as input to a feed forward network with linear elements to learn the B matrix that produce estimates $\hat{x}$ of x.

Sanger's rule learns weights $a_{i,j}$ so that the output $y_i = \sum_{j=0}^{11} a_{i,j}x_j$. The algorithm begins with any initial A and updates weights according to $\Delta a_{i,j} = \eta y_i(x_j - \sum_{k=0}^{i} y_k a_{i,j})$, where η is a small learning rate that determines how quickly A changes. The FFN begins with an initial B and learns weights that produce output $\hat{x} = By$. The update rule for B is $\Delta b_{i,j} = \eta(x_i - \hat{x}_i)y_j$.

The matrix A need not be stored with the compressed image as it is not needed for decompression. However, the matrix B must be stored as part of the compressed image so that the $\hat{x}$ (decompressed image) can be produced from y (the decompressed reduced image). Each element of B can be adequately stored in 4 bytes, hence it takes $4*4*12 = 192$ bytes to store B. We also want to store the mean vector, μ_{12}, which requires 12 bytes (1 byte per component). We store only four principal component multipliers for each 2×2 pixel block and use 8 bits for the first principal component multiplier and 6 bits for multipliers 2, 3, and 4. Hence, we need 26 bits for each 2×2 block. Therefore, for a 256×256 image we need $192(8) + 12(8) + (26)(128)^2 = 427{,}616$ bits. This compares with 1,572,864 bits for the original image. Hence, we obtain a compression ratio of 3.7.

The arrangement of the compressed y components in the reduced image in preparation for the subsequent application of a gray-scale compression algorithm is important to the overall results. There are four y components and one should arrange each component in a different quadrant of the image plane. Fig. 2 depicts the four principal components of a reduced color image with components arranged in different image quadrants. This arrangement eliminates artificially induced high-frequency content that would result if all four components of y were arranged adjacent to each other. High-frequency content makes gray-scale compression more difficult. As it can be seen from Fig. 2, each y component is in general quite different from the other y components, while each individual y component displays high-local spatial correlation.

3. Results

In this section we present results using each of YIQ, YUV and the PCA/BP hybrid learning as color reduction methods on the images shown in Fig. 3. Note that these images are color images but are only shown here in black and white. For YIQ reduction, we first transform the RGB image to the YIQ representation, using the best bit allocation rate found empirically, 6, 5, and 5 bits for each of the Y, I, and Q components, respectively, and then use the inverse transform to convert back to the RGB representation. We perform the same operations for the YUV representation, using 6, 5, and 5 bits, respec-

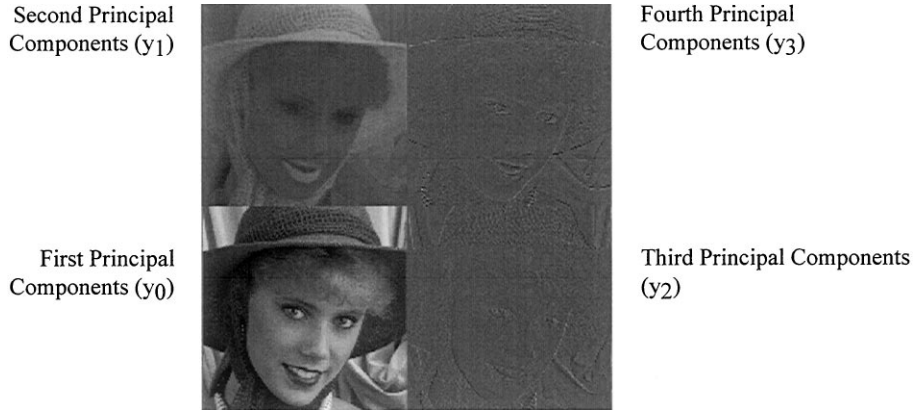

Fig. 2. Reduced color image.

Table 1
Color compression results

	YIQ				YUV				PCA/BP Learning			
Picture	R	G	B	C	R	G	B	C	R	G	B	C
Lady	29.4	41.3	31.0	**31.7**	31.5	41.3	26.7	30.1	32.2	32.3	35.6	33.1
Baboon	28.2	41.1	32.8	**31.5**	30.9	39.5	27.3	30.3	34.6	34.7	35.5	34.9
Boy	29.3	41.6	31.3	**31.8**	30.9	40.8	27.0	30.2	31.5	31.7	34.7	32.4
Cave	28.2	40.2	33.8	**31.7**	31.4	40.9	27.8	30.9	31.7	36.1	32.8	33.2
Chalk	28.9	40.0	30.7	**31.3**	29.3	41.0	27.1	29.7	34.9	35.7	34.8	35.1
Coral	29.3	39.4	31.6	**31.8**	30.6	40.0	27.8	30.6	29.5	28.5	30.2	29.3

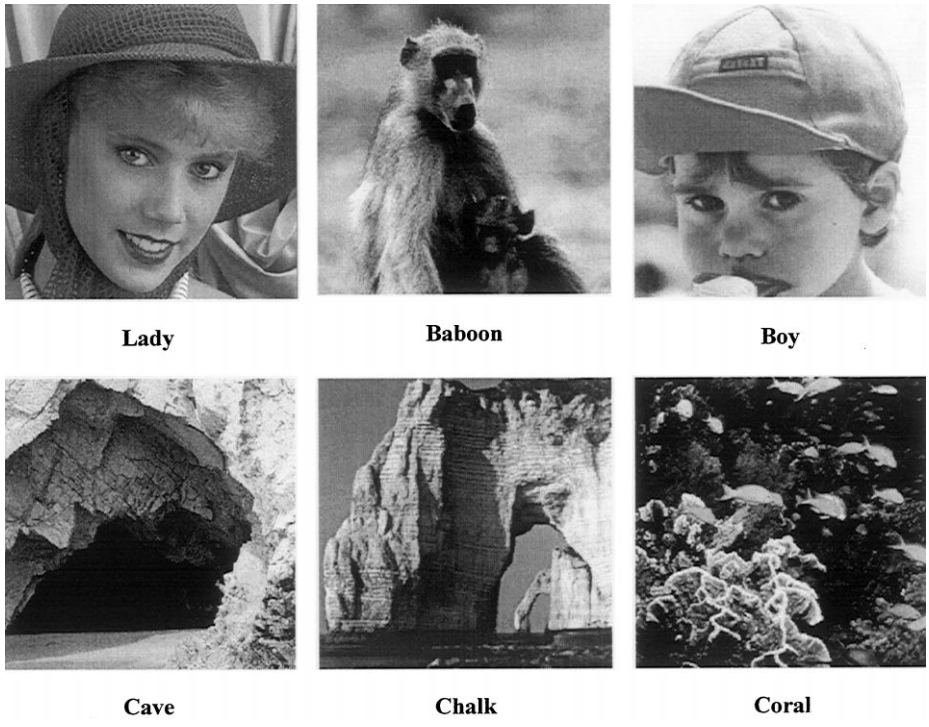

Fig. 3. Test images.

tively, for Y, U, and V components. For each of YIQ and YUV we obtain a compression ratio of 1.5.[1] For the learning approach, we learn the transform weights as well as inverse transform weights as described above using a value of 1.0E-6 for the learning rate and use 20,000 repetitions. Learning takes about 9 s on a 133 MHz Pentium processor. We convert each image to the principal component representation as in Fig. 2 and use 8, 6, 6, and 6 bits, respectively, for each of the first four principal components. As explained above, this bit allocation results in a compression ration of 3.7. Next we use the weights for the inverse transform to convert back to the RGB representation. For each image and each color reduction method, we compare the original RGB image

[1] $(3)(8)/(6 + 5 + 5) = 1.5$

Table 2
Progress of learning

	Step	Picture	R	G	B	C	Rep
1.	Train	Chalk	9.3/25.6	9.3/26.9	7.3/25.5	8.5/26.0	0/1000
2.	Train	Cave	23.9/23.5	25.1/25.7	24.4/24.4	24.4/24.4	1000/2000
3.	Train	Chalk	25.0/25.8	27.3/27.0	25.0/26.9	25.6/26.5	2000/3000
4.	Train	Cave	24.4/25.7	25.9/27.2	25.4/26.8	25.2/26.5	3000/4000
5.	Train	Chalk	25.4/27.3	26.4/29.0	28.2/29.3	26.5/28.4	4000/5000
6.	**Test**	**Boy**	**29.5**	**22.7**	**29.3**	**25.9**	**5000**
7.	**Test**	**Lady**	**29.5**	**23.5**	**28.4**	**26.3**	**5000**
8.	Train	Cave	26.2/25.4	27.4/28.1	27.6/28.6	27.0/27.1	5000/6000
9.	Train	Chalk	25.2/26.8	28.7/30.8	30.0/32.2	27.5/29.3	6000/7000
10.	**Test**	**Boy**	**30.4**	**23.9**	**28.9**	**26.8**	**7000**
11.	**Test**	**Lady**	**28.9**	**24.2**	**28.6**	**26.7**	**7000**
12.	Train	Cave	25.5/27.0	28.4/29.6	29.7/30.8	27.4/28.8	7000/8000
13.	Train	Chalk	27.3/31.4	31.5/34.1	31.8/34.5	29.8/33.1	8000/9000
14.	**Test**	**Boy**	**31.0**	**24.4**	**29.1**	**27.2**	**9000**
15.	**Test**	**Lady**	**30.5**	**24.7**	**28.9**	**27.3**	**9000**

with the resulting RGB image produced from the reduced image. We calculate four PSNR[2] values for each image in order to make comparisons. These four values correspond to the noise induced in red (R), green (G), blue (B), and combined (C) channels. Results are shown in Table 1.

In Table 1, R, B, G, and C columns hold PSNR values for red, green, blue and combined RGB, respectively. In five of six cases, we see that learning outperforms YIQ and YUV with respect to PSNR, even though in all six cases, learning achieves more than twice the compression as the other two methods. The performance of learning is due to two factors. First, the learning method adapts to individual images where YIQ and YUV apply globally without change to all images. Second, the learning method takes advantage of spatial correlation of color in images which the YIQ and YUV methods do not.

The primary disadvantage of the learning approach is the computation time required for learning. Once learning has taken place, the conversion to the reduced image and back to the new RGB image is very fast. Hence, if the results of learning for one image could be applied to another image, significant improvement in overall compression time would result. To show that learning from one image does apply to other images we try a second experiment in which we train on the Chalk and Cave images in Fig. 3 and apply training results to the Boy and Lady images. Table 2 below shows PSNR results for this experiment where normal rows correspond to training exemplars and bold rows correspond to test images. Learning using the training examples proceeds in steps of 1000 repetitions (approximately 0.5 s) instead of 20,000 repetitions as used for results in Table 2. The learning rate is again 1.0E-6. In cells where two PSNR values are depicted (e.g. 9.3/25.6), PSNR values are for training images and the first number is the PSNR just before training is performed and the second number is the PSNR just after training is performed. In cells where only one PSNR value is shown, the PSNR is for a test image and no training was conducted at that step. The Rep column shows the number of cumulative training repetitions before and after training at that step. Hence, two numbers are shown for the training exemplars and one number for the test images. Step 1 shows the results for the chalk image when the weights are initialized to random values between -1.0 and 1.0.

Table 2 shows that as learning proceeds on the training images, that reproduced image quality of the training images improves, as would be expected. However, we also see that the image quality of the test images improves, even though the test images are not used for training. This experiment shows that it is feasible to

[2] PSNR is Peak Signal to Noise Ratio. PSNR $= 20 \log_{10}(d/\mathrm{RMS})$ where d is the peak signal which is usually 255 and RMS is the root-mean-square error between the original image component and the estimated image component. For each of red, green and blue PSNR calculations,

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(p_i - \hat{p}_i)^2}$$

where N is the number of pixels in the image, p_i is the corresponding red, green, or blue value of the ith pixel in the original image and $\hat{p}_i$ is the estimate of p_i. The combined channel RMS, denoted $\mathrm{RMS}_c = \sqrt{\frac{1}{3}(\mathrm{RMS}_r^2 + \mathrm{RMS}_g^2 + \mathrm{RMS}_b^2)}$ where RMS_r, RMS_g, and RMS_b are the RMS for red, green and blue components, respectively.

apply training from one image to previously unseen images.

4. Conclusion

This paper introduces a hybrid learning method consisting of PCA and backpropagation for reducing ('compressing') an RGB color image to a black and white image with the same number of pixels as the original color image. This reduction achieves a 3.7 compression ratio while retaining the capability of reproducing an approximation of the original color image that is optimal with respect to the minimum square error. The resulting black and white image can be further compressed using a gray-scale compression method such as JPEG. Experimental results show that as our hybrid learning method adapts to local (spatial) image characteristics it outperforms the YIQ and YUV standard compression methods. Our experiments also show that it is feasible to apply training results from one image to previously unseen images.

References

[1] D.B. Judd, D.L. McAdam, G. Wyszecki, Spectral distribution of typical daylight as a function of correlated color temperature, J. Opt. Soc. Amer. A 54 (1964) 1031–1040.

[2] Y. Fisher (Ed.), Fractal Image Compression, Theory and Application, Springer, New York, NY, 1995.

[3] M. Nelson, The Data Compression Book, M & T Books, San Mateo, CA, 1982.

[4] Healey, E. Glenn, A.S. Steven, B.W. Lawrence, Physics Based Vision: Principles and Practice: Color Vol. 2, A.K. Peters Ltd., 1992.

[5] D.B. Judd, G. Wyszecki, Color in Business, Science, and Industry, Wiley, New York, 1975.

[6] F. Grum, C.J. Bartleson (Eds.), Optical Radiation Measurements, Vol. 2, Academic Press, New York, 1980.

[7] T.D. Sanger, Optimal unsupervised learning in a single-layer linear feedforward neural network, Neural Networks, (2) (1989) 459–473.

[8] J. Hertz, K. Anders, G.P. Richard, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, MA, 1991.

About the Author—HARRY WECHSLER received the Ph.D. in Computer Science from the University of California, Irvine, in 1975, and he is presently Professor of Computer Science at George Mason University. His research on intelligent systems has been in the areas of PERCEPTION: Computer Vision, Automatic Target Recognition, Signal and Image Processing, MACHINE INTELLIGENCE: Pattern Recognition, Neural Networks, Machine Learning (ML), Information Retrieval, Data Mining and Knowledge Discovery, EVOLUTIONARY COMPUTATION: Genetic Algorithms and Animats, MULTIMEDIA and VIDEO PROCESSING: Large Image Databases, Document Processing, and HUMAN-COMPUTER INTELLIGENT INTERACTION: Face and Hand Gesture Recognition, Biometrics and Forensics. He was Director for the NATO Advanced Study Institutes (ASI) on "Active Perception and Robot Vision" (Maratea, Italy, 1989), "From Statistics to Neural Networks" (Les Arcs, France, 1993), "Face Recognition: From Theory to Applications" (Stirling, UK, 1997), and he served as co-Chair for the International Conference on Pattern Recognition held in Vienna, Austria, in 1996. He authored over 200 scientific papers, his book "Computational Vision" was published by Academic Press in 1990, he is the editor for "Neural Networks for Perception" (Vols. 1 & 2), published by Academic Press in 1991, and co-editor for "Face Recognition: From Theory to Applications", published by Springer-Verlag in 1998. He was elected as an IEEE Fellow in 1992 and as an Int. Assoc. of Pattern Recognition (IAPR) Fellow in 1998.

About the Author—CLIFFORD CLAUSEN received the Ph.D. in Information Technology from George Mason University, Fairfax, VA, in 1999. Currently, Clifford works for Unisys corporation as a software engineer. His research interests include image compression, signal processing, computer vision, neural networks, pattern recognition, medical information systems, information retrieval, learning systems, and data mining.