Vector Quantization by Principal Component Analysis

Yeong-Wei Chen and Chaur-Chin Chen *
Department of Computer Science
National Tsing Hua University
Hsinchu 300, Taiwan
e-mail: cchen@cs.nthu.edu.tw

Abstract

An adaptive vector quantization (VQ) scheme based on principal component analysis is proposed. Each codevector in the codebook is the centroid of training vectors belonging to the leaf of a binary tree constructed based on a tree classifier using the first principal components of training vectors. A simple thresholding method on the first principal components used to find the split key in each nonterminal node. Our scheme can encode an image as soon as the codebook is constructed. Experiments on a Pentium 150 PC with 32MB RAM under Chinese Windows 95 environments show that the proposed scheme requires, in average, 2.5 seconds to encode a 512×512 image with a codebook of size 256 to achieve 0.625 bits per pixel (bpp) with the average PSNR value exceeding 30 for four images *Lenna, Jet, Peppers*, and *Scene*. A comparison with three other commonly used VQ algorithms is also given.

1 Introduction

Vector quantization for image compression has been widely studied since Linde, Buzo, and Gray published their pioneer work [13]. The work can be divided into three parts as codebook design, encoding procedure and decoding procedure. An image is first partitioned into 4×4 nonoverlapping blocks which are represented as 16-tuple vectors called training vectors. In LBG algorithm [13], given the codebook size N, for example, N=256, a K-means clustering algorithm is applied to partition the training vectors into N clusters of which the centroid of each cluster is called a codevector in the codebook. Training a codebook of size 256 using a 512×512 image requires around 4 minutes running on a Pentium 150 PC with 32MB RAM under Windows 95. To encode a 512×512 image by using an exhaustive search, we have to compute $16384 \times 256 \times 16 \times 3$ multiplications and additions which requires 8 seconds on a Pentium 150 PC. Image decoding is just block paste which can be done in a second. A variety of schemes have been proposed to improve the codebook design, either to speed up the generation of the codebook or increase the fidelity of a reconstructed image. Whereas, a universally best codebook would have never existed. The evaluation of a codebook can only be done by experiments.

This paper addresses a new scheme of VQ for image compression based on principal component analysis (PCA) [1] and a tree classifier [5]. For a given image, our scheme is adaptive and can encode this image simultaneously as the codebook is constructed. The time for a codebook generation with encoding requires only 2.5 seconds cpu time for a 512×512 gray scale image on a Pentium 150 PC with 32MB RAM under Windows 95 environments.

The remaining of this paper is organized as follows. Section 2 reviews three commonly used VQ schemes. Section 3 proposes a new VQ scheme called PCA-TSVQ based on principal component analysis and tree structure. Section 4 shows experimental comparisons for PCA-TSVQ with LBG, fast PNN, and DCT-TSVQ schemes. Section 5 gives the conclusion.

2 VQ schemes: a review

The fundamental idea of VQ for image compression is to establish a codebook consisted of codevectors such that each codevector can represent a group of image blocks of size 4×4 to achieve the goal of storage reduction. An image or a set of images is first partitioned into 4×4 nonoverlapping blocks which are represented as 16-tuple vectors called training vectors. The size of training vectors can be very large. For example, a 512×512 image contributes 16384 training vectors. The goal of codebook design is to establish a few representa-

*This work was partially supported by NSC 86-2213-E-007-051

tive vectors called codevectors, say 256 or 512, from a set of training vectors. The encoding procedure is just finding the index of the codebook corresponding to the codevector which is *closest* to a given block. Many methods have been reported to generate a codebook. Nasrabadi and King [14] gave a good review. In this paper, we review LBG [13], fast PNN [6], and DCT-TSVQ [9] algorithms for a comparison with our proposed scheme described in the next section.

2.1 LBG algorithm

The LBG algorithm [13] applied a K-means clustering algorithm [11] to design a codebook from a set of training images. An LBG algorithm is listed below.

LBG Algorithm

1. Input training vectors $\{x_i\}$.
2. Initialize a codebook $C = \{y_j; 1 \leq j \leq N\}$, set $m = 0$, $D_0 = \infty$.
3. Partition the training vectors $\{x_i\}$ into N clusters by $x_i \in S_p$ if $d(x_i, y_p) \leq d(x_i, y_j)$ for all j, where d is a 2-norm. Set $m \leftarrow m + 1$ and compute $D_m = \sum_{p=1}^{N} \sum_{x_i \in S_p} d(x_i, y_p)$.
4. Update each centroid y_j by $y_j = \frac{1}{|S_p|} \sum_{x_i \in S_p} x_i$.
5. If $(D_{m-1} - D_m)/D_m > \epsilon$, goto step 3.
6. The codebook consists of the codevectors y_j, $1 \leq j \leq N$.

Using an LBG algorithm to establish a codebook is time consuming. Both fast PNN [6] and DCT-based [9] algorithms are proposed to speed up the time for codebook generation.

2.2 Fast PNN algorithm

The idea of pairwise nearest neighbor (PNN) algorithm [6] starts with the clusters of the entire training vectors, each cluster consists of a single training vector. The algorithm successively merges two closest clusters into a new cluster represented by the centroid of training vectors belonging to these two clusters and the process iterates until a desired size of clusters is achieved. The centroid of each cluster corresponds to a codevector, a codebook is thus established. The PNN algorithm generally takes much more time than an LBG one to generate a codebook. However, Equitz [6] proposed using a K-d tree structure together with the strategy of PNN to reduce the time for a codebook generation which is referred to as a *fast PNN* algorithm. Assuming that all of the training vectors are put in the root. The fast PNN algorithm allocates the training vectors in each nonterminal node into the two children by using the mean (or median) of a single component of the training vectors with maximum variance as the key. A binary tree is thus constructed with only a few training vectors sitting in the leaves called buckets. The algorithm merges the nearest pair of training vectors in the same bucket and rebalances the K-d tree repeatedly until the desired size of codebook is obtained. Compared with the LBG algorithm, a fast PNN algorithm usually saves more than 90% of the time to generate a codebook using a single training image.

2.3 DCT-based algorithm

Motivated by the fast PNN algorithm, Hsieh et al. [9] proposes a VQ scheme using a binary tree structure and discrete cosine transform. We denote this scheme as DCT-TSVQ scheme. The algorithm starts with putting all of the training vectors into the root node of a binary tree. It successively splits the training vectors in a nonterminal node into its two children by using the mean of DCT coefficients [18] with the maximum variance as a split key. The procedure continues until the desired number of leaves which corresponds to the codebook size is achieved. The codebook is the collection of the centroids of training vectors in the corresponding leaves.

2.4 Summary

A codebook generated by one of the LBG, fast PNN, or DCT-based algorithms can be used for encoding images. To avoid an exhaustive search of a closest codevector for any given vector, some fast search techniques [2] [17] may be used to accelerate the encoding time.

3 PCA-TSVQ scheme

Motivated by pattern recognition techniques [5], an *efficient algorithm*, PCA-TSVQ [4], based on principal component analysis (PCA) [1] and tree structure is proposed in this section. The scheme is adaptive and can encode an image as soon as the codebook is established. The training images are first partitioned into 4×4 nonoverlapping blocks which are represented as 16-tuple vectors called

training vectors and are put into the root node in the beginning. The algorithm is listed below.

PCA-TSVQ Algorithm

1. For training vectors in each nonterminal node, compute the corresponding first principal components [1].
2. Apply Otsu's thresholding method [16] on the principal components to assign training vectors to the two children of that node.
3. The above processes are repeated until the number of leaves matches the number of codebook size.
4. The centroid of each leaf is a codevector in the codebook.
5. The training vectors in the same leaf can be assigned the same index and are represented as the centroid of that leaf in real-time applications.

The first principal components of training vectors in each nonterminal are the projections of those vectors on the direction of eigenvector corresponding to the largest eigenvalue of the covariance matrix estimated by those training vectors. The *power method* [8] can be applied to find the eigenvectors. Although the PCA-TSVQ algorithm listed above may not have minimum distortion between encoded vectors and codevectors, it can save the extra time for encoding. Experiments in next section show that little image fidelity is lost. More principal components may be used to improve the fidelity of decoded images which is another issue.

4 Experimental comparison

Some experiments for designing codebooks based on a PCA-TSVQ algorithm are compared with those designed based on LBG, fast PNN, and DCT-TSVQ algorithms. All programs were written in a visual C++ code implemented on a Pentium 150 PC with 32MB RAM under Chinese Windows 95 environments. All images reported in this paper have size 512×512 and the codebook size is restricted to be 256. For each comparison, we report the CPU time (in seconds) for generating a codebook and the peak signal-to-noise ratio (PSNR) between an original image and the corresponding image decoded from compression. The fast decoding time [2] for a 512×512 image using a codebook of size 256 requires about $2 \sim 3$ seconds CPU time.

Image	Lenna	Jet	Peppers	Scene
LBG	31.41	31.00	29.57	27.91
fast PNN	30.46	29.14	28.89	27.29
DCT-TSVQ	30.61	30.11	29.10	27.61
PCA-TSVQ	30.74	30.73	29.54	27.93

Table 1: PSNR values using Baboon, Lenna, and Jet as training images

4.1 Experiment 1

Using three images: Barboon, Lenna, and Jet [15] as training images to design codebooks by LBG, fast PNN, DCT-TSVQ, and PCA-TSVQ schemes requires 1056 seconds, 29 seconds, 8 seconds, and 9 seconds, respectively. Table 1 shows the PSNR values when decoding for four images, two are inside the training set and the other two are outside the training set. The results indicate that our codebook generation time is much faster than traditional LBG and fast PNN algorithms and 1 second slower than that of DCT-TSVQ. However, the fidelity of decoded images based our scheme are consistently better than DCT-TSVQ and fast PNN schemes and approximates that by the LBG scheme.

4.2 Experiment 2

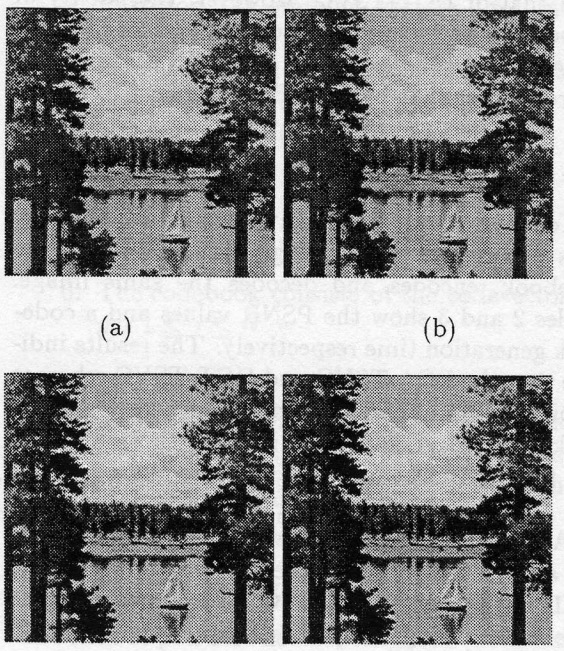
To be practical, an adaptive method is preferred. This experiment uses a single image to train the codebook, encodes and decodes the same image. Tables 2 and 3 show the PSNR values and a codebook generation time respectively. The results indicate that the PCA-TSVQ and DCT-TSVQ schemes require no more than 3 seconds to generate a codebook with image fidelity as good as those by LBG and fast PNN schemes. The decoded images of *Jet* and *Scene* using LBG, fast PNN, DCT-TSVQ, and PCA-TSVQ algorithms shown in Figures 1 (a~d) are all visually close to the original image. The DCT-TSVQ and PCA-TSVQ algorithms take less time for encoding. Thus, the adaptive PCA-TSVQ and DCT-TSVQ may be considered as practical real-time compression algorithms for the up-to-date PCs and faster computers with the request of a moderate compression ratio.

4.3 Experiment 3

This experiment compares two adaptive schemes, PCA-TSVQ and DCT-TSVQ with a suboptimal encoding strategy. We assign the training vectors belonging to the same leaf the index corresponding

Image	Lenna	Jet	Peppers	Scene
LBG	32.39	31.16	31.37	28.74
fast PNN	31.68	30.95	31.32	28.37
DCT-TSVQ	31.43	30.87	31.15	28.36
PCA-TSVQ	32.06	31.30	31.35	28.75

Table 2: PSNR values using a single training image



(a) (b)

(c) (d)

Figure 1: Decoded Scene by (a) LBG, (b) fast PNN, (c) DCT-TSVQ, and (d) PCA-TSVQ algorithms

Image	Lenna	Jet	Peppers	Scene
LBG	285	292	292	293
fast PNN	4.12	4.01	3.90	4.28
DCT-TSVQ	1.92	1.81	1.82	1.75
PCA-TSVQ	2.58	2.52	2.52	2.53

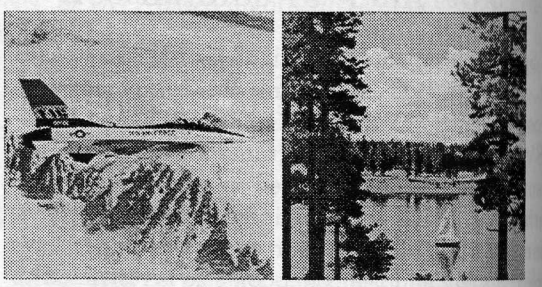
Table 3: CPU time for a codebook generation using a single training image

Image	Lenna	Jet	Peppers	Scene
DCT-TSVQ	31.02	30.27	30.28	27.99
PCA-TSVQ	31.74	30.86	30.96	28.46

Table 4: PSNR values when encoding as a codebook is generated

to the centroid of that leaf so that an extra encoding procedure can be skipped to save computation time. That is, the encoding is done as soon as the codebook is generated. The encoding times are listed in Table 3. The PSNR values shown in Table 4 are very close to the corresponding parts listed in Table 2. Whereas, the proposed PCA-TSVQ scheme consistently has larger PSNR values than those by DCT-TSVQ scheme. Figure 2 shows the decoded images by the PCA-TSVQ suboptimal encoding strategy.

Experiments show that our proposed PCA-TSVQ scheme with a suboptimal encoding strategy is feasibl in real-time compression to achieve a moderate compression ratio.



(a) (b)

Figure 2: Decoded Jet and Scene by the PCA-TSVQ algorithm using suboptimal encoding strategy

5 Conclusion

Compared with transform coding such as JPEG-based method using discrete cosine transform [12], Fractal-based method [10] [7], and Wavelet-based method [19] [20]. the vector quantization does not necessarily requantize the coefficients via bit allocation. Instead, VQ encodes images as the indices of codevectors, where each codevector is the mean vector of a subset of training vectors. The design of VQ is much simpler than transform coding. The traditional LBG and fast PNN schemes require too much time in encoding which may not be practical. Our proposed PCA-TSVQ takes less than 3 seconds on a Pentium 150 PC to encode a 512×512 image to achieve compression ratio 0.625 bits per pixel with a satisfactory decoded image quality. The strategy of VQ can be slightly modified to select a color palette [3] with few (e.g., 256) representative colors among 16.7 million colors. We conclude that PCA-TSVQ scheme with a suboptimal encoding strategy may be used as a practical real-time compression for gray level images under a request of compression ratio around 16.

References

[1] T. W. Anderson (1984). An introduction to multivariate statistical analysis. Wiley, New York.

[2] C. Bei and R. M. Gray (1985). An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Trans. on Comm.* 33, 1132-1133.

[3] J.P. Braquelaire and L. Brun, Comparison and optimization of methods of color image quantization, *IEEE Trans. on Image Processing*, vol. 6, 1048-1052, 1997.

[4] Y.W. Chen, Vector Quantization by Principal Component Analysis, *M.S. Thesis*, Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, June, 1998.

[5] P.A. Devijver and J. Kittler (1982). Pattern Recognition: A statistical approach. Prentice-Hall.

[6] W.H. Equitz (1989). A new vector quantization algorithm. *IEEE Trans. on ASSP* 37, 1568-1575.

[7] Y. Fisher Editor, Fractal Image Compression: Theory and Applications, Springer-Verlag, 1995.

[8] G.H. Golub and F. Van Loan, Matrix computations (1989). *The Johns Hopkins University Press.*

[9] C.H. Hsieh, P.C. Lu, and J.C. Chang (1992). DCT-based codebook design for vector quantization of images. *IEEE Trans. Circuits and Systems for Video Techonology* 2, 401-409.

[10] A.E. Jacquin (1992). Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing* 1, 18-30.

[11] A.K. Jain and R.C. Dubes (1988). Algorithms for clustering data. Prentice-Hall.

[12] A.K. Jain (1989). Fundamentals of digital image processing. Prentice-Hall.

[13] Y. Linde, A. Buzo, R. M. Gray (1980). An algorithm for vector quantizer design, *IEEE Trans. on Comm.* 36, 84-95.

[14] N.M. Nasrabadi and R.A. King (1988). Image coding using vector quantization: a review. *IEEE Transactions on Comm.* 36, 957-971.

[15] A.N. Netravali and B.G. Haskell (1989). Digital Pictures: Representation and Compression. Plenum Press.

[16] N. Otsu (1979). A threshold selection method from gray level histograms. *IEEE Trans. on Systems, Man, and Cybernetics* 9, 62-66.

[17] J.S. Pan, F.R. McInnes, and M.A. Jack (1996). Fast clustering algorithms for vector quantization. *Pattern Recognition* 29, 511-518.

[18] W.B. Pennebaker and J.L. Mitchell (1993). JPEG still image data compression standard. *Von Nostrand Reinhold.*

[19] J.M. Shapiro, Embedded image coding using zerotree of wavelet coefficients, *IEEE Trans. on Signal Processing*, vol. 41, 3445-3462, 1993.

[20] A. Said and W.A. Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, 243-250, 1996.