Adaptive hierarchical vector quantization for image coding: new results

Sheng Zhong
University of Hong Kong
Department of Computer Science
Hong Kong
and
Peking University
Department of Mathematics
Beijing, China
E-mail: zhsh@csd.hku.hk

Francis Chin
University of Hong Kong
Department of Computer Science
Hong Kong

Qing Yun Shi
Peking University
Center for Information Science
Beijing, China

Abstract. Adaptive hierarchical algorithms of vector quantization (VQ) for image coding are proposed. First the basic codebook is generated adaptively using adaptive VQ, then the quadruplets of codes/indices in the so-called zigzag order are coded into higher level (second and third levels) codes by creating the second- and third-level index codebooks to reduce the redundancy presented in the codes. Partially matched quadruplets are also encoded in the second and third layers using the index codebooks along with corresponding correction schemes. The third-layer encoding achieves a better compression ratio than a two-layer encoding scheme, which was shown to be optimal when partial encoding was not adopted. This three-layer coding scheme achieves better compression with no extra distortion and little extra computation. Experiments show encouraging results.

Subject terms: vector quantization; index codebook; partial encoding; three-layer encoding.

Optical Engineering 34(10), 2912–2917 (October 1995).

1 Introduction

Vector quantization (VQ) techniques have been widely used in the area of image coding because of their ability to achieve a low bit rate.[1–4] VQ techniques achieve a low bit rate by exploiting the correlation and redundancy between blocks.

Nonadaptive VQ techniques match a vector with the fixed codebook, which has been generated by training over a large set of training vectors from different images. This type of technique, however, suffers from two major problems. One is its expensive computational time. The other is its poor ability for generalization, i.e., its poor performance when used to process images outside the training set. The latter problem becomes worse especially where intensity edges occur. Thus a number of methods have been designed to solve this problem.[5–8] Adaptive VQ (AVQ) techniques[9,10] have also been proposed. Adaptive VQ does not suffer much of the two previous problems because a much smaller codebook with smaller code vectors can be designed for each input image. Nevertheless, the compression ratio thus achieved by an adaptive VQ is usually lower than that achieved by nonadaptive VQ because of the smaller vector dimension and the side information that needs to be transmitted or stored.

Nasrabadi[11] and Vaisey and Gersho[12] adopted a variable block size scheme to achieve lower bit rate. They divided the image into blocks of different sizes according to the quadtree representation. The small blocks are coded by a typical VQ or adaptive VQ, whereas the large blocks are coded by a transform VQ to discard the high-frequency coefficients. Zhong et al.[13] proposed an adaptive hierarchical vector quantization (AHVQ) method in which high-layer codebooks were designed adaptively, making use of the redundancy in larger areas in images. For example, in the second layer, quadruplets of codes/indices from the basic adaptive VQ were encoded by the second-layer index codebook when they appeared in the index codebook. The second-layer encoding lowers the bit rate significantly without causing any more distortion, and requires little additional computation. This AHVQ scheme is found to be able to achieve good results when a two-layer structure is adopted.[13]

Paper 16114 received Nov. 10, 1994; revised manuscript received March 8, 1995; accepted for publication March 8, 1995.

In this paper, we present two new hierarchical AVQ schemes based on AHVQ. The improved AHVQ (IAHVQ) algorithm further utilizes the second-layer codebook by including a partial encoding scheme. The partial encoding scheme of IAHVQ changes the optimal structure of adaptive hierarchical VQ given in Ref. 13 because the third-layer encoding process can be included. The optical structure of this new three-layer AHVQ (NAHVQ) is found to be more efficient than AHVQ and IAHVQ.

The rest of the paper is arranged as follows. The AHVQ algorithm is briefly described in Sec. 2. In Sec. 3, the IAHVQ method is described. The three-layer scheme, NAHVQ, is given in Sec. 4. Experimental results and comparisons with other methods are shown in Sec. 5. The discussion and conclusions are given in Sec. 6.

2 Adaptive Hierarchical Vector Quantization

In AVQ techniques, an $N \times N$ image is first divided into a set of small blocks/vectors (e.g., of size 2×2). Then the set is used to design a small codebook by using the traditional VQ techniques such as the algorithm Linde-Buzo-Gray. Each block from the set is matched to the best code vector in the codebook and is represented by the index of that code vector. Then, both the codebook and the indices are transmitted. At the receiver, the reconstruction of the image is rather straightforward. It is just a searching process in the codebook. Here the adaptivity is on a picture-by-picture basis, i.e., each picture has its own codebook. Thus improved reconstruction quality can be obtained compared with nonadaptive VQ. Other variations of this scheme first divide the image into several regions and then use the preceding scheme in each region, or in the transformed domain.

After the codebook and codes/indices are generated, the statistics of the indices are examined. Usually the neighboring pixels in an image are highly similar and correlated. So are the neighboring blocks. They are therefore spatially redundant. The combinations of the neighboring blocks can be expected to present some regularity, which can be used to further eliminate the redundancy between the neighboring blocks.

AHVQ (Ref. 13) makes use of the indices of the small blocks (called basic blocks) encoded by the AVQ. These indices form a digital map that we call the index map of the original image, as in Fig. 1(a).

Let A, B, C, and D be the indices of the four basic blocks in the upper left corner. Consider the vector (A,B,C,D). It actually represents the appearance of the upper left large block. If such a block of (A,B,C,D) occurs frequently in other positions in the image, it is worth coding it as a single code and treating the quadruple vector (A,B,C,D) as a code vector in the new codebook, which we call the index codebook. We explain its generation below.

Let us first regulate a scanning order in the index map. As illustrated in Fig. 1(b), we scan the indices in a zigzag order, which is called Z order. The Z order is quite similar to that of the quadtree representations,[14] especially the depth-first representation. The Z order forms the data structure of the coded data. It provides an efficient representation for hierarchical coding. It is adopted also because it makes it easy to employ similarities between large blocks in the image in a hierarchical way.

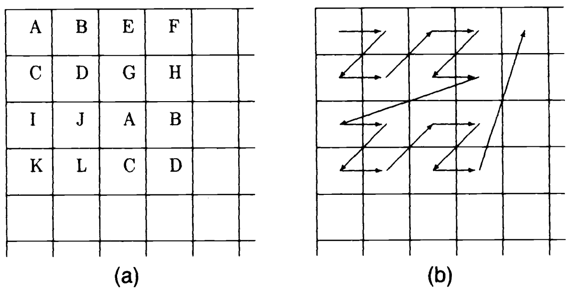


Fig. 1 (a) Index map and (b) Z order.

Now we count the occurrences of each index quadruple vector (A,B,C,D) in the index map according to the Z order. Each of the vectors (A,B,C,D) forms a larger block of the quadtree representation that corresponds to a larger block in the original image. We choose the L (a positive integer, usually a power of 2 for the sake of being coded by bits) vectors of (A,B,C,D) that occur most frequently to be the representative vectors and form the index codebook. The index map derived from the adaptive VQ now undergoes the encoding process. It is examined in the Z order to see whether or not four neighboring indices, which form a larger block in the original image, can match a code vector in the index codebook. That encoding process can be concurrently done in the process of selecting the second-layer code vectors by keeping track of the addresses of the quadruplets that contribute to the code vectors. Thus, the computational time for the second-layer coding is minor compared to that for the basic layer's VQ coding. In addition, the second-layer coding is lossless.

Note that not all the indices in the index map can be grouped to form a vector in the index codebook. The more the code vectors in the index codebook, the more the indices/codes in the index map that can be grouped into a higher level code. On the other hand, both the code length of the index code vectors and the side information, i.e., the index codebook that should be transmitted will increase. Thus, there is a trade-off between the overall bit rate and the size of the index codebook.

The same process can be done again. The indices derived from the index codebook can further be examined to form a higher level—the third-layer index codebook. This process can be carried out until the desired level is reached. The index codebooks thus added and the hierarchical structure are the side information to be transmitted. There is again a trade-off between the overall bit rate and the hierarchical level.

The experiments in Ref. 13 show that the best hierarchical scheme should include only one layer of index coding, which means that the codes/indices from the basic VQ coding will only be further coded by the second-layer codebook. This is because only a few of the indices from the second-layer coding can be further coded as a third-layer code, and thus, the savings of bits cannot counteract the increase of side information if a third layer is included; or even if it can, the gain is quite small.

It is also shown in Ref. 13 that for most of the images we tested, the bit rates are minimum when the size of the index codebook of the second layer is four times as large as that of the basic AVQ codebook. Usually the bit rate can be lowered significantly if the AHVQ is used in the preceding manner. For example, for the basic image blocks of size 2×2

and basic codebook size to be of length 32, the index codebook is of length 128. Using the fast adaptive and partial distortion (FAPSPD) algorithm[15] as the basic adaptive VQ(AVQ), AHVQ lowers the bit rate from 1.270 bits/pixel of the adaptive VQ to 0.910 bits/pixel for the 256×256 256-gray-level "Lenna" image; the compression ratio is improved by nearly 40%. And the subjective quality of the reconstructed image is quite good.

3 Improved AHVQ

In the AHVQ method just introduced, in addition to the side information of the second-layer codebook that needs to be transmitted or stored, 1 bit must be added to each quadruplet to tell whether the quadruplet is encoded by a second-layer codeword or not. The further compression achieved by the hierarchical scheme is determined by the percentage of the quadruplets that are encoded by the second-layer codebook. Let's look at a simple analysis using the example given at the end of the last section.

Suppose that the total number of the quadruplets is *'number of quad'* and

$b * \textit{number of quad}$

quadruplets are encoded by the second-layer codebook, where $0 \leq b \leq 1$; then

$d * \textit{number of quad}$

quadruplets are left uncoded by the second-layer codebook, where $d = 1 - b$.

Because each basic code/index in the index map takes 5 bits and each index-code takes 7 bits, the savings of bits when a quadruplet is encoded by the second-layer codebook are $4 \times 5 - (7 + 1) = 12$ bits. Thus, the total savings are

$(12 * b - d) * \textit{number of quad}$

bits. When $b < d/12$, AHVQ expands instead of compressing data. Fortunately, in practice, we have never met such an image. This is because there exist many similarities in most images.

Now, let us look for ways to further utilize the second-layer codebook. In the second-layer encoding process, we find many such quadruplets that have only three of their components matched (called 3-matched) with those of a second-layer codeword but can not be further encoded by AHVQ. For example, Fig. 2(a) is a large block in the Z order, corresponding to a 8×8 block in the original image. Suppose the upper right subblock '(A,B,C,D)', where A, B, C, and D are basic codebook indices, can only be 3-matched. This kind of quadruplet can be encoded by a second-layer codeword (say 'b') with the mismatched component corrected by coding its position in the quadruplet (which needs 2 bits) and followed by its first-layer code (which needs 5 bits) in the basic codebook [Fig. 2(b)]. For the sake of simplicity, we call these quadruplets 3-matched coded (partially encoded), where the fully matched quadruplets are 4-matched coded (fully encoded). We calculate the bit saving as follows.

Suppose *'c * number of quad'* quadruplets among the quadruplets that are not fully encoded by the second-layer codebook (there are *'d * number of quad'* such quadruplets) are 3-match coded. Then only *'a * number of quad'* quadruplets are left neither 4-matched coded nor 3-matched coded by the second-layer codebook, where $a = d - c$ and now $a + b + c = 1$. Three identifiers of 1 bit or 2 bits, {0,10,11}, are used to distinguish these three types of quadruplets. Because c is usually much smaller than a and b, c is coded by a 2-bit identifier, whereas the shorter 1-bit identifier 0 is used to distinguish the larger type of a and b. The saving of each 3-matched coding is $4 \times 5 + 1 - (2 + 7 + 5 + 2) = 5$ (bits). Then the savings of bits achieved by partial encoding over AHVQ are

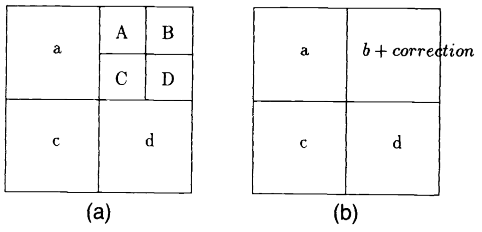


Fig. 2 (a) and (b) Partial matching.

$[5c - \min(a,b)] * \textit{number of quad}$ bits.

The $-\min(a,b)$ means that an extra bit has been used to identify the smaller type of a and b and should be subtracted from the savings.

Further compression over AHVQ is achieved when

$$c > \frac{\min(a,b)}{5} ,$$

which is true in all of our experiments on natural images. Thus we call the preceding coding scheme the IAHVQ.

Because the search for the 3-matched pattern in the index codebook is largely done in the search process for the 4-matched pattern, the additional computation for 3-match coding is quite small. In addition, the coding is lossless.

In a similar way, the 3-match-encoding technique can be extended to the 2-match-encoding technique in the second-layer codebook with a correction scheme of the two mismatched components. In the case where the basic codebook size = 32 and the second-layer codebook size = 128, it is not worth coding the 2-matched quadruplets because no positive gain will be obtained.

In the case where the basic codebook size = 64 and the second-layer codebook size = 64*4 = 256, IAHVQ can achieve further compression over AHVQ by

$[7c - \min(a,b)] * \textit{number of quad}$ bits ,

which has greater potential of compression than the case where the basic codebook size = 32 and the second-layer codebook size = 128.

Dixit and Feng[16] proposed a hierarchical VQ method in which both the basic VQ coding and second-layer index coding process are nonadaptive and the basic layer codebook and the second-layer index codebook were obtained by training over a large sequence of images, and in which only the fully match coding scheme was used. The reported results with regard to the design of the second-layer codebook and determination of the parameters are quite similar to those in the AHVQ (Ref. 13). In such nonadpative schemes where both the basic VQ codebook and second-layer codebook are much larger than in AHVQ, 512 and 2048, respectively, for example, greater compression can be achieved if a scheme like IAHVQ is adopted. The savings of bits by IAHVQ over AHVQ are expressed as

$$[13c - \min(a,b)] * \textit{number of quad} \text{ bits} \ .$$

4 Three-Layer Scheme: NAHVQ

After the 3-matched quaduplets are coded, the structure of the hierarchical coding scheme may be somewhat different from that of the AHVQ in Ref. 13 because the 3-match-coded quadruplets in the second layer increase the possibility of grouping four neighbor second-layer codes into a third-layer code. For example, Fig. 2(a) is a large block in the Z order, corresponding to an 8×8 block in the original image. Suppose three subblocks in it can be encoded by the second-layer codebook as a, c, and d. The upright subblock '(A,B,C,D)' can only be 3-match coded as the form '$b + correction$'. See Fig. 2(b) for an illustration. In the AHVQ method in Ref. 13, the large block remains as '$a + (A,B,C,D) + c + d$', unable to be encoded by a third-layer codeword. However, by the IAHVQ introduced above, the large block appears as '$a + (b + correction) + c + d$'; thus, it might be encoded into a third-layer code 'β' if '(a,b,c,d)' forms a codeword 'β' in the third-layer codebook, just by coding the position of the (A,B,C,D) subblock followed by the '*correction*' term. In fact, the idea of 3-match coding in the IAHVQ method can be generalized to the third-layer encoding even if the upright subblock '(A,B,C,D)' cannot be 3-match coded in the second layer through encoding the whole block as '$\beta + positioncode + A + B + C + D$'.

In the original work[13] on AHVQ, the structure of AHVQ is determined to be two layered as the optimal because with the 4-match coding scheme, many second-layer indices in the Z order are not consecutive and thus only a few quadruplets of the second-layer indices can be grouped into third-layer indices according to the Z order. The 3-match-coding scheme proposed in the previous section, however, increases this possibility as just stated. Moreover, the idea of 3-match coding can be extended to the third-layer encoding, i.e., a quadruplet of the second-layer indices can be encoded even if it only matches a codeword in the third-layer codebook partially as long as an appropriate correcting scheme exists. All these factors may make a three-layer encoding structure work better than the two-layer structure in AHVQ and IAHVQ.

We specify the three-layer (NAHVQ) coding algorithm in the following manner:

1. The basic AVQ encoding is done as in AHVQ (Ref. 13).
2. The second-layer codebook is generated as in AHVQ and partial encoding is done as stated in Sec. 3.
3. The third-layer codebook of certain size is generated by selecting the quadruplets with most frequent occurrences in the second-layer index map according to the Z order; and the third-layer encoding is done as follows: for the quadruplets of the second-layer indices in the Z order,
   a. whose four components are perfect second-layer indices and match a third-layer codeword, encode them as the third-layer codeword (pattern 1: 4-matched encoding)
   b. whose three components are perfect second-layer indices and one component is a 3-matched index in the second layer, i.e., an index like '$b + correction$' as previously described, and match a third-layer codeword, encode them as the third-layer codeword followed by encoding the position of the '$b + correction$' term and the '*correction*' term itself (pattern 2: 4-matched encoding + correction)
   c. whose four components are perfect second-layer indices and only three components match the correspondences in a third-layer codeword, encode them as the third-layer codeword followed by encoding the position of the mismatched component and the index of it (pattern 3: 3-matched encoding + correction1)
   d. whose three components are perfect second-layer indices and one component is a 3-matched index in the second-layer, i.e., an index like '$b + correction$' as previously described, and only three second-layer indices match the correspondences in a third-layer codeword, encode them as the third-layer codeword followed by encoding the position of the mismatched component and the second-layer index of it, then followed by encoding the position of the '$b + correction$' term and the '*correction*' term itself (pattern 4: 3-matched encoding + correction2)
   e. whose three components are perfect second-layer indices and one component is a quadruplet of the first-layer indices, i.e., of the form '(A,B,C,D)', and the three second-layer indices match the correspondences in a third-layer codeword, encode them as the third-layer codeword followed by encoding the position of the '(A,B,C,D)' term and the first-layer indices of $A, B, C,$ and D, respectively (pattern 5: 3-matched encoding + correction3).

In the preceding NAHVQ algorithm, only the five third-layer encoding patterns are included because a careful study of the statistical property shows that those five patterns are the frequently occurring patterns and each encoding of them will save many bits. For example, with the three-layer structure of sizes (32, 128, 32), each encoding of those five patterns will save 23, 20, 13, 10, and 12 bits, respectively. Because of the different frequencies of the five matching patterns, the identifiers for them can be designed using static Huffman coding.

The larger the third-layer codebook, the more the quadruplets of the second-layer indices that will be encoded into third-layer indices. However, the side information, e.g., the third-layer codebook, will increase too. So there is a tradeoff between the size of the third-layer codebook and the compression ratio achieved. A fourth-layer encoding process can be similarly established; however, usually no positive gain can be obtained because few quadruplets of the third-layer indices can be coded further.

5 Experiments

We tested the IAHVQ and NAHVQ algorithms as well as the AVQ and AHVQ algorithms for comparisons. For the basic codebook generation, we adopt the FAPSPD method.[15] We set the basic block size to be 2×2 and the basic codebook size to be 32; the index codebook size is set to be of size 128.

First, we determine the optimal size of the third-layer codebook in NAHVQ. We only consider the sizes of 0, 2, 4, 8, 16, 32, 64, ... We find the optimal size for the third-layer codebook to be around 16; for most cases 16 is exactly the optimal third-layer codebook size. Figure 3 is the experimental results for one of the tested images: the "Lenna" image of size 256×256 and 256 gray levels.

Next, we show the superiority of IAHVQ and the three-layer encoding scheme NAHVQ to other two-layer methods such as AHVQ by comparing NAHVQ with them with the third-layer codebook size fixed at 16.

Figure 4(a) is the original image of size 256×256 256-gray-level monochrome image of "Zelda." Figure 4(b) shows the reconstructed image [note that the reconstructed images using AVQ, AHVQ, IAHVQ, or NAHVQ are all the same as Fig. 4(b) because index coding is lossless]. Table 1 presents some experimental results.

We find that NAHVQ achieves the best results among the listed methods; IAHVQ also achieves very good results. Compared with AVQ, the compression ratio is improved by about 65.19% using NAHVQ and by about 53.01% using IAHVQ, whereas by only about 47.47% using AHVQ. The improvements are obtained by the third-layer encoding and the partial encoding.

The computational cost of IAHVQ and NAHVQ is almost the same as that of AHVQ. The additional computation used for the second-layer partial encoding and the third-layer encoding is negligible compared with that for the basic AVQ encoding in the first layer.

The results of the tests on the "Cronkite" and "Lena" images of size 256×256, 256 gray levels are presented in Figs. 5 and 6 and Tables 2 and 3. Again, we find that NAHVQ gets better results than AVQ, AHVQ, and IAHVQ. Actually, with dozens of images that we have tested, NAHVQ always performs better than AVQ, AHVQ, and IAHVQ. IAHVQ also shows improved performance.

6 Discussion and Conclusions

We have designed two new adaptive hierarchical vector quantization techniques, IAHVQ and NAHVQ. They are superior to the two-layered encoding scheme, AHVQ, which is optimal when full encoding is used. The partial encoding of the quadruplets in the Z order has changed the optimal structure of the hierarchical coding scheme.

Compared with the single-layer AVQ and the two-layered AHVQ algorithms, the index-coding process of NAHVQ or IAHVQ has been shown to be effective and worthwhile for the following reasons:

1. It lowers the bit rate significantly.
2. It requires little additional computational time.
3. It causes no more distortion.

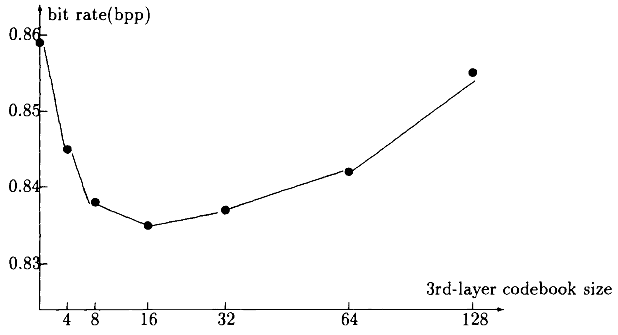


Fig. 3 Optimal third-layer codebook size.

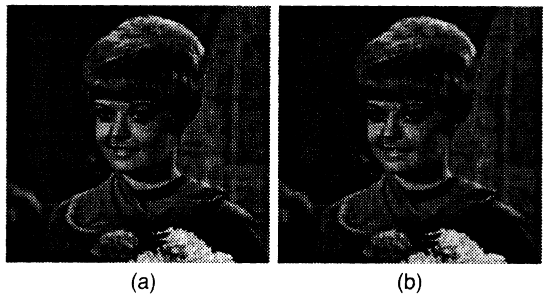


Fig. 4 "Zelda": (a) original and (b) reconstructed.

Table 1 Experiments on "Zelda."

algorithms	AVQ	AHVQ	IAHVQ	NAHVQ
a	100%	37.84%	22.04%	-
b	-	62.16%	62.16%	-
c	-	-	15.80%	-
bit rate	1.270	.863	.827	.766
comp. ratio	6.32: 1	9.32: 1	9.67: 1	10.44: 1
PSNR	30.6 dB	30.6 dB	30.6 dB	30.6 dB

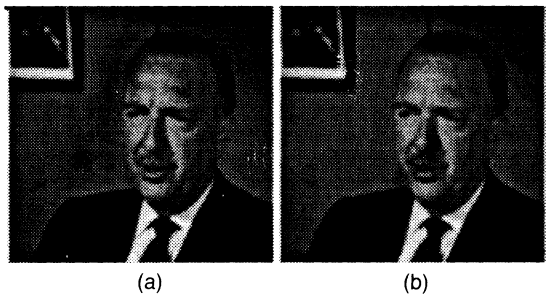


Fig. 5 "Cronkite": (a) original and (b) reconstructed.

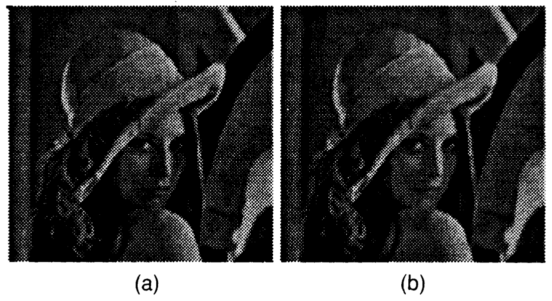


Fig. 6 "Lena": (a) original and (b) reconstructed.

Table 2 Experiments on "Cronkite."

algorithms	AVQ	AHVQ	IAHVQ	NAHVQ
a	100%	17.14%	10.77%	-
b	-	82.86%	82.86%	-
c	-	-	6.37%	-
bit rate(bpp)	1.270	.711	.681	.525
comp. ratio	6.32: 1	11.24: 1	11.74	15.25: 1
PSNR	32.0 dB	32.0 dB	32.0	32.0 dB

Table 3 Experiments on "Lena."

algorithms	AVQ	AHVQ	IAHVQ	NAHVQ
a	100%	43.60%	22.78%	-
b	-	56.40%	56.40%	-
c	-	-	20.82%	-
bit rate(bpp)	1.270	.910	.859	.835
comp. ratio	6.32: 1	8.79: 1	9.32: 1	9.58: 1
PSNR	31.7 dB	31.7 dB	31.7 dB	31.7 dB

Compared with the AHVQ algorithm,[13] the second-layer codebooks of NAHVQ and IAHVQ are further utilized to encode those partially matched quadruplets and thus redundancy is further reduced. Furthermore, NAHVQ includes a third-layer encoding process that is very effective, whereas IAHVQ adopts only a two-layer structure.

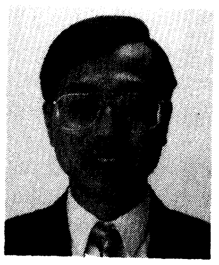
The idea of partial encoding has much greater potential when applied to nonadaptive VQ where the basic codebook is of much larger size. A fixed nonadaptive index codebook can also be designed for an adaptive VQ, or an adaptive index codebook can be designed for a nonadaptive VQ. The fixed index codebook should be generated by training over a large set of index maps and stored in both the transmitter and the receiver so that the side information no longer needs to be transmitted.

In addition, in the encoding of NAHVQ or IAHVQ, we find that the more the index quadruplets (A, B, C, D) (including the 3-matched ones) occur in the index codebook, the lower the bit rate will be. Also the effects of compression achieved by the 4-matched encoding and 3-matched encoding are different. Using (32, 128, 16) structure as an example, in the second-layer codebook design, three 3-matched encodings can save 15 bits, which is larger than that saved by one 4-matched encoding. Therefore, a more efficient or even optimal way to select the second-layer and the third-layer codebooks can be further studied.

References

1. Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.* **COM-28,** 84–95 (Jan. 1980).
2. A. Gersho and B. Ramamurthi, "Image coding using vector quantization," in *Proc. IEEE Int. Conf. Acoust. Speech Image Proc.*, pp. 428–431 (May 1982).
3. N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: a review," *IEEE Trans. Commun.* **COM-36**(8), 957–971 (1988).
4. R. L. Baker and R. M. Gray, "Image compression using nonadaptive spatial vector quantization," in *Proc. 16th Asilomar Conf. Circuits, Systems and Computers* (Oct. 1982).
5. R. L. Baker and R. M. Gray, "Differential vector quantization of achromatic imagery," in *Proc. Int. Picture Coding Symp.*, pp. 55–61 (1982).
6. R. A. King and N. M. Nasrabadi, "Image coding using VQ in the transform domain," *Pattern Recogn. Lett.* **1,** 323–329 (1983).
7. B. Ramamurthi and A. Gersho, "Classified vector quantization of images," *IEEE Trans. Commun.* **COM-34**(11), 1105–1115 (1986).
8. D. S. Kim and S. U. Lee, "Image vector quantizer based on a classification in the DCT domain," *IEEE Trans. Commun.* **COM-39**(4), 549–556 (1991).
9. M. Goldberg, P. R. Bouder, and S. Shilen, "Image compression using adaptive vector quantization," *IEEE Trans. Commun.* **COM-34**(2), 180–187 (1986).
10. A. Habibi, "Survey of adaptive image coding techniques," *IEEE Trans. Commun.* **COM-25,** 1275–1284 (Nov. 1977).
11. N. M. Nasrabadi, "Use of vector quantizers in image coding," in *Proc. IEEE Int. Conf. Acoust. Speech Image Proc.*, pp. 125–128 (Mar. 1985).
12. D. J. Vaisey and A. Gersho, "Variable blocksize image coding," in *Proc. IEEE Int. Conf. Acoust. Speech Image Proc.*, pp. 1051–1054 (Apr. 1987).
13. S. Zhong, Q. Y. Shi, and M. T. Cheng, "Adaptive hierarchical vector quantization for image coding," *Pattern Recogn. Lett.* **15**(12), 1171–1175 (1994).
14. H. Samet, "The quadtree and related hierarchical data structures," *ACM Comput. Surveys* **16**(2), 188–260 (1984).
15. S. Zhong, Q. Y. Shi, and M. T. Cheng, "Fast adaptive partial search and partial distortion algorithm for VQ," in *Proc. Int. Conf. on Neural Networks and Signal Processing,* (Nov. 1993), Guangzhou, China, South China University Press.
16. S. S. Dixit and Y. Feng, "Hierarchical address vector quantization for image coding," *CVGIP: Graph. Mod. Image Process.* **53**(1), 63–70 (1991).

Sheng Zhong received his BSc from the Department of Mathematics at Peking University in 1988, his MSc from the Center of Information Science at Peking University in 1990, and his PhD from the Department of Mathematics at Peking University in 1994. He is now a senior research associate in the Department of Computer Science at University of Hong Kong as well as a lecturer in the Department of Mathematics at Peking University. His research interests involve image processing, image and video coding, computer vision, pattern recognition, wavelet transform and neural networks. Dr. Zhong has published more than 20 refereed papers.



Francis Chin received his BASc from the University of Toronto and his MSc, MA, and PhD degrees from Princeton University. He is now a professor and chairs the Department of Computer Science at the University of Hong Kong. He is a senior member of IEEE. His main research interests include algorithms, computational geometry, image processing, and computer vision. He has served on the program committees of numerous international workshops and conferences.

Qing Yun Shi is a professor in the Center of Information Science at Peking University and directs the Academic Committee of the National Laboratory on Machine Perception. She is a member of the governing board of the International Association of Pattern Recognition and of the Chinese Academy of Sciences. Her main research areas include image processing, pattern recognition, computer vision, and image database systems.