A VARIANT OF ADDRESS VECTOR QUANTIZATION FOR IMAGE COMPRESSION USING LOSSLESS CONDITIONAL ENTROPY CODING

Wenshiung Chen
Dept. of Electrical Engineering
Feng Chia University

En-Hui Yang and Zhen Zhang
Dept. of Electrical Engineering-Systems
University of Southern California

ABSTRACT

In this paper, a variant of address vector quantization (ADVQ) algorithm for image compression using conditional entropy lossless coding is presented. The motivation of the proposed approach is derived from Shannon's basic entropy concept that conditional entropy is less than joint entropy.

1. INTRODUCTION

Recently, data compression (or source coding), especially for image and video, has boomed due to the demands in some applications, such as HDTV and multimedia. There has been many coding techniques developed in the past two decades. Vector quantization (VQ) has been found to be an efficient coding technique due to its inherent ability to exploit the high correlation between the neighboring pixels. Some excellent survey articles and books are given in [3]. VQ is widely used in image/video and speech compression applications, because simple table look-up encoding and decoding procedures may be used.

Many VQ coding techniques which exploit the interblock correlation have been developed [3]. Researchers have introduced finite state VQ (FSVQ) [2], Adaptive VQ (AVQ) with a dynamically refining codebook, called the Gold-Washing (GW) algorithm [1], address VQ (ADVQ) [5] [6], and so forth. Information theory claims that conditioning reduces entropy. Based on this, a new variant of address VQ for image compression is introduced which exploits the interblock correlation of the statistical redundancy between the blocks via the conditional probability. The lower bit rate is obtained without introducing any extra coding distortion so that the coding scheme can be viewed as a lossless coding technique.

2. REVIEW OF ADDRESS VQ

Address VQ [5] is a combination of a standard VQ technique with a lossless coding technique. The basic idea of the ADVQ coding method is based on the joint entropy concept.

However, the ADVQ coding method has some disadvantages. One of them is the computational complexity of computing the score parameters in the encoder and reordering the address-codebook in the encoder and the decoder. Another one is the necessity of transmitting the statistics of reordering. In [5], they ignored the statistics transmission. Due to the use of 0.06 bits/pixel for this purpose, the performance gain should be $0.437/0.316 = 1.38$ (or 72.31%) rather than 1.70.

3. A VARIANT OF ADDRESS VQ

Address VQ is a block code while our coding scheme can be viewed as a conditional code. The basic idea of the new variant of ADVQ is based on exploiting the strong interblock correlation as well as the statistical redundancy via the knowledge of the conditional probabilities. Information theory claims that conditioning reduces entropy; that is, $H(\mathbf{X}|\mathbf{Y}) \leq H(\mathbf{X})$, where $\mathbf{X}$ and $\mathbf{Y}$ are two random variables. Intuitively, the theorem says that $\mathbf{Y}$ can reduce the bit rate for coding $\mathbf{X}$. The encoding process consists of two phases: lossy coding followed by lossless coding. First, the coded image is first decomposed into 4×4 sub-image blocks. Each 4×4 block is then coded block-by-block sequentially in the order from top to bottom, and from left to right by using VQ matching in the LBG-codebook. The result of this phase is a sequence of integer index numbers. In the next phase, a lossless coding on the set of the index numbers is performed. This phase does not affect the resulting distortion produced in the first phase. In this algorithm, we utilize the conditional probability which is viewed as a priori knowledge. The knowledge of the previously coded blocks is utilized for coding the current block. The idea for coding is described as follows in detail.

Suppose that $\{X_{ij}^k\}_{i=1\,j=1}^{n\quad n}$ is the 2-D array of blocks to be coded and $\{I_{ij}\}_{i=1\,j=1}^{n\quad n}$ is the array of the associated indexes. Three indexes of the previously coded blocks as the prior conditions are utilized to code the current block. In geometric terms, $X_{i,j-1}$ is the symbol preceding the "current" symbol $X_{i,j}$, $X_{i-1,j}$ is the symbol above $X_{i,j}$ in a 2-D raster scan of the image, and $X_{i-1,j-1}$ is the symbol nearest to $X_{i,j}$ in the upper left direction. These three symbols can be expected to have a strong influence on the current symbol $X_{i,j}$. Since the three conditions are known at the encoder and the decoder, no statistics transmission is needed. Assume X_1, X_2, and X_3 have already been coded and represent the upper-left (diagonal) neighboring block, upper (vertical) neighboring block, and left (horizontal) neighboring block, with respect to the currently coded block X_4, respectively (Fig. 1(a)). Let I_1, I_2, I_3, and I_4 be the indexes which are associated with blocks X_1, X_2, X_3, and X_4, respectively. Specifically, there are four types of index-matching used for coding the current block, as shown in Fig. 1. In Fig. 1(c), type B utilizes I_2 and I_3 only. Whereas, type C and type

0-7803-2431-5/95 $4.00

D, as shown in Fig. 1(d) and Fig. 1(e), respectively, utilize only I_2 or I_3.

The proposed algorithm consists of two codebooks: a LGB-codebook $\mathcal{C}_{\mathrm{LBG}}$ and four distinct and independent 2-stage index-codebooks $\mathcal{C}_{\mathrm{A}}, \mathcal{C}_{\mathrm{B}}, \mathcal{C}_{\mathrm{C}}$, and $\mathcal{C}_{\mathrm{D}}$ which are associated with type A, type B, type C, and type D, respectively. The basic structure of the index-codebook is a 2-stage codebook, (Fig. 2(a)), where the first stage is called the "*primary index-codebook*" denoted by $\mathcal{C}_{\mathrm{P}}$, and the second is called the "*secondary index-codebook*" denoted by $\mathcal{C}_{\mathrm{S}}$. In our design, the $\mathcal{C}_{\mathrm{P}}$ is a large codebook with size N_{P}. The codevector in the $\mathcal{C}_{\mathrm{P}}$ is made up of two fields: the first field is an index-codevector which contains k_{iv} indexes which may be any specific combination of I_{i1}, I_{i2} and I_{i3}, where k_{iv} is the dimension of the index-codevector which depends on what type of index matching is used. Obviously, $k_{iv} = 3, 2, 1$, or 1 for type A, B, C, or D, respectively. The second field is a pointer which points to the $\mathcal{C}_{\mathrm{S}}$. All of the $\mathcal{C}_{\mathrm{S}}$'s are small codebook, with the variable sizes from 1 to N_{S}, in which each vector contains only one index component I_{i4}.

The four types of index-matching operations and their corresponding index-codebooks are in detail described as follows:

- *Type A: 3-Index Matching.* In this type, three prior indexes are pre-matched (see Fig. 1(b)). The index-codevector is made up of 3 indexes, I_{i1}, I_{i2}, and I_{i3}; i.e.,$\vec{I}_i = (I_{i1}, I_{i2}, I_{i3})$ (Fig. 2(a)). The encoding operation (or matching criterion) is as follows. If the three indexes I_1, I_2, and I_3 are totally matched with i-th index-codevector with the index components I_{i1}, I_{i2}, and I_{i3} in the $\mathcal{C}_{\mathrm{PA}}$, then prior condition is matched. The procedure goes through the pointer into the corresponding $\mathcal{C}_{\mathrm{SA}}$. If the index I_4 is matched with one of the codevectors I_{i4} in $\mathcal{C}_{\mathrm{S}}$, then index matching occurs. This means that a variable-length code rather than index code, is used to code the current block X_4.

- *Type B: 2-Index Matching.* Type B utilizes two prior indexes for pre-matching (see Fig. 1(c)). The structure of the $\mathcal{C}_{\mathrm{B}}$ is similar to that of type A except $\vec{I}_i = (I_{i2}, I_{i3})$. The encoding operation is almost the same as in type A but only two indexes in the primary index-codebook are compared.

- *Type C: Vertical-Index Matching.* Type C utilizes only one prior index, which is the index of the upper neighboring block, for pre-matching (see Fig. 1(d)). Similarly, the structure of the $\mathcal{C}_{\mathrm{C}}$ is similar to that of type A except $\vec{I}_i = (I_{i2})$.

- *Type D: Horizontal-Index Matching.* Type D also utilizes only one prior index, which is the index of the left neighboring block, for pre-matching (see Fig. 1(e)). The structure of the $\mathcal{C}_{\mathrm{D}}$ is similar to that in type A except $\vec{I}_i = (I_{i3})$.

The flowchart of the encoding process is shown in Fig. 3.

4. INDEX-CODEBOOK DESIGN

4.1. Index-codebook Design

The index-codebooks are generated off-line during the training process. We extract all the possible four-index combinations of the four neighboring blocks occurring together in the 8×8 blocks. The entries in the index-codebook are called the index-codevectors.

In the simulation, only those of the index combinations whose occurrence frequency is larger than a threshold value are sieved as the index-codevectors in the index-codebook. As a result of the high correlation between these small blocks, the sizes of the N_P, would be much smaller than the total number of possible index combinations. For example, if the LBG-codebook size is $N = 128$, and the dimension of the codevectors in the primary index-codebook (for type A) is $k_{iv} = 3$ and the dimension of the codevectors in the secondary index-codebook is 1, then the total number of possible combinations is $N^{(k_{iv}+1)} = 128^{(3+1)} = 268,435,456$. However, the index-codebook obtained by the training data is much smaller than this value.

In ADVQ, reordering address-codebook is an on-line operation. It is advantageous that there is no reordering operation for the index-codebook needed in our algorithm. Thus, the processing time for reordering is saved. It is possible to limit the size of the index-codebook to a desired size by only retaining the most probable block (or index) combinations, and thus to reduce the computational complexity.

A simple data structure for constructing the index-codebook is sequential table. The procedure first searches the table by fully-matching criterion until the entry whose index-codevector I_{i1}, I_{i2}, and I_{i3} are totally matched to the input index combination I_1, I_2, and I_3 is met, and then go into the corresponding secondary index-codebook associated with the entry. Thus, it is important that reducing the computational complexity for searching in the primary index-codebook.

4.2. Tree Structure for Reducing Time Complexity

One efficient way for reducing the complexity of the index match searching is to adopt a 4-level tree structure for $\mathcal{C}_{\mathrm{A}}$, as shown in Fig. 2(b). The upper 3 levels belong to primary index-codebook. The lowest level belongs to secondary index-codebook. On the first level, only one node exists and is composed of N fields where each contains a pointer which may point to either *null* or a same type node on the second level. For each node on the second level, it is the same as on the first level. For each node on the third level, it may point to either *null* or a secondary index-codebook with variable size.

The number of operations for searching primary index-codebook is at most 3 index integer comparisons and 3 pointer pass operations. Similarly, the number of index integer comparison operations for searching secondary index-codebook is proportional to the size of the codebook (e.g., maximum is only 5 in our simulation). Thus, at most 8 operations of index integer comparison is sufficient for lossless index matching. Explicitly, the extra computational complexity caused by lossless index matching is very low. Similarly, a 3-level tree with the same structure as $\mathcal{C}_{\mathrm{A}}$ is constructed for $\mathcal{C}_{\mathrm{B}}$ and two 2-level trees are constructed for $\mathcal{C}_{\mathrm{C}}$ and $\mathcal{C}_{\mathrm{D}}$.

5. EXPERIMENTAL RESULTS

In order to compare the coding fidelity among the different techniques, the peak signal-to-noise ra-

Table 1: Simulation results

	Standard VQ		M/RVQ		ADVQ (PM/RVQ)		Outside		Inside	
Image	Lena	peppers	Lena	peppers	Lena	peppers	Lena	peppers	Lena	peppers
Block Size	4 X 4	4 X 4	4 X 4	4 X 4	4 X 4	4 X 4	4 X 4	4 X 4	4 X 4	4 X 4
N_{LBG}	128	128	128	128	128	128	128	128	128	128
N_{PA} (Type A)	—	—	—	—	—	—	10059	10059	80035	80035
N_{PB} (Type B)	—	—	—	—	—	—	10087	10087	68055	68055
N_{PC} (Type C)	—	—	—	—	—	—	163	163	822	822
N_{PD} (Type D)	—	—	—	—	—	—	166	166	830	830
bpp	0.437	0.437	—	—	0.316	0.320	0.248	0.236	0.105	0.106
CR	18.28	18.28	—	—	25.32	25.00	32.25	33.80	75.87	75.44
MSE	93.00	141.00	47.00	80.00	56.88	87.54	55.24	85.36	55.24	85.36
PSNR	28.45	26.64	31.41	29.11	30.58	28.71	30.71	28.82	30.71	28.82

tio (PSNR) and mean-squared-error (MSE) are used

5.1. LBG-Codebook Design

We have extended PM/RVQ [5] to a fine-tuned version that utilizes more preceding coded context and extrapolation to predict a more precise estimation for the mean. Let m_{h1} and m_{h2} be the mean of four left-most and right-most pixels of the block X_3, m_{v1} and m_{v2} be the mean of four upper-most and lower-most pixels of the block X_2, and x_{d1} and x_{d2} be grey level of the upper-left pixel and the lower-right pixel of the block X_1. The estimated mean of the block X_4 is computed by

$$\begin{aligned}\bar{m} &= \frac{1}{18}((-m_{h1}+3m_{h2}+-m_{v1}+3m_{v2})\cdot 4 \\ &+ (-x_{d1}+3x_{d2})).\end{aligned}$$

The LBG-codebook is designed by using the well-known LBG algorithm [4]. The size of the LBG-codebook is $N = 128$ (7 bits).

5.2. Index-Codebook Design

The index-codebooks are off-line designed during the training session. 18 training images have been used to count the occurrence frequency of each index-codevector. Those index combinations that the occurrence frequency is more than the threshold value 1 have been extracted to constitute $\mathcal{C}_A$ and $\mathcal{C}_B$. Those index combinations that the occurrence frequency is more than the threshold value 100 have been extracted to constitute $\mathcal{C}_C$ and $\mathcal{C}_D$. In the simulation, $N_{PA} = 10,059$, $N_{PB} = 10,087$, $N_{PC} = 163$, and $N_{PD} = 166$. Since the number of bits used for coding each block by LBG-codebook is 7 bits only, this implies that the size of the secondary index-codebooks has to be limited and not more than 5. Hence, we select the following parameters: $N_{SAi} \leq 5, N_{SBi} \leq 5, N_{SCi} \leq 4$, and $N_{SDi} \leq 4$, for each $i, 1 \leq i \leq N_P$. In the secondary index-codebook, a variable-length code like Huffman code is designed and the set of codes is shown in Fig. 2(a).

5.3. Experimental Results

For $k = 4 \times 4 = 16$, there will be a total number of 16,384 blocks needed to be coded. The test images are "Lena" and "peppers". For "Lena", we can evaluate $R = 0.248$ bits/pixel (compression ratio, CR = 32.25). Similarly, we can also evaluate R= 0.236 bits/pixel (CR = 33.80) for "peppers". The summary of the simulation results is shown in Table 1. The coding bit reduction are $0.248/0.437 = 56.75\%$ for "Lena" and $0.236/0.437 = 54\%$ for "peppers", respectively. In other words, the performance gain of our algorithm are 1.95 for "Lena" and 1.98 for "peppers" with respect to the standard VQ's. The computational complexity is much lower than that of ADVQ algorithm. The constructed image are shown in Fig. 5(a) and Fig. 5(b).

6. CONCLUSION

In this paper, we have introduced a variant of the ADVQ, where interblock correlation and conditional entropy concept are exploited to reduce the bit rate below what is possibly achieved by a standard memoryless VQ without any extra distortion. We conclude that the proposed algorithm is superior to the ADVQ in performance and complexity.

References

[1] W. S. Chen, Z. Zhang, and E.-H. Yang. A hybrid adaptive vector quantizer for image coding via Gold-Washing mechanism. submitted to *IEEE Trans. on Image Processing*, 1994.

[2] J. Foster, R. M. Gray, and M. Ostendorf Dunham. Finite-state vector quantization for waveform coding. *IEEE Trans. on Information Theory*, IT-31(5):348–359, May 1985.

[3] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwar Academic Publishers, 1992.

[4] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, COM-28(1):84–95, Jan. 1980.

[5] N. M. Nasrabadi and Y. Feng. Image compression using address-vector quantization. *IEEE Trans. on Communications*, COM-38(12):2166–2173, Dec. 1990.

[6] N. M. Nasrabadi and Y. Feng. A multilayer address vector quantization technique. *IEEE Trans. on Circuits and Systems*, CAS-37(7):912–921, Jul. 1990.

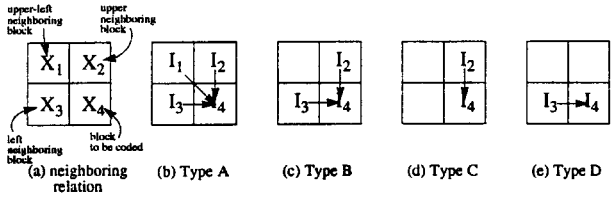

Figure 1: Four types of index matching.

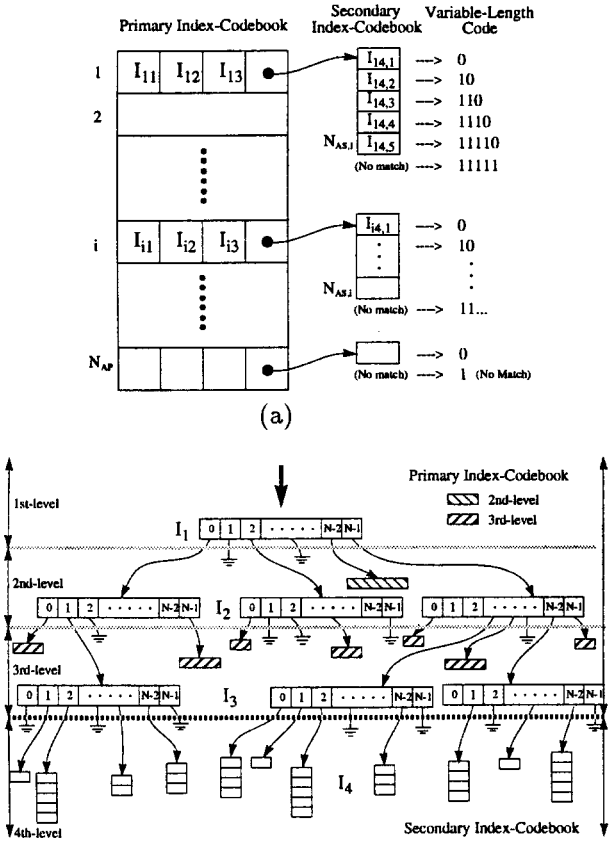

Figure 2: (a) The structure of the index-codebook (for the case of type A only). (b) Tree structure of the index-codebook (for the case of type A only).

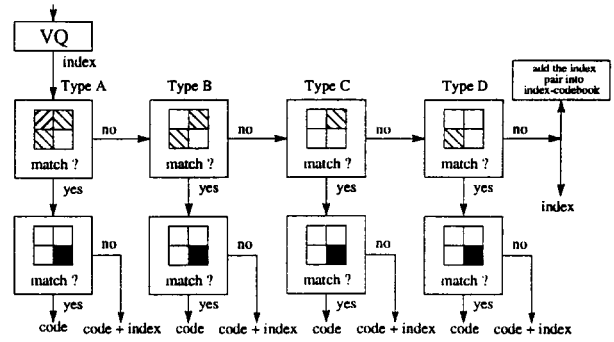

Figure 3: Flowchart of the variant ADVQ encoding

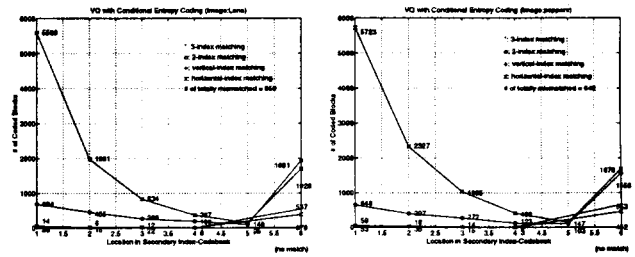

Figure 4: Number of blocks coded by the LBG-codebook and the index-codebooks. (a) "Lena." (b) "peppers."

Figure 5: The constructed images. (a) "Lena": bpp=0.248, MSE=55.24. (b) "peppers": bpp=0.236, MSE=85.36.