International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:7, 2008

Modified Vector Quantization Method for Image Compression

K.Somasundaram, and S.Domnic

***Abstract*—**A low bit rate still image compression scheme by compressing the indices of Vector Quantization (VQ) and generating residual codebook is proposed. The indices of VQ are compressed by exploiting correlation among image blocks, which reduces the bit per index. A residual codebook similar to VQ codebook is generated that represents the distortion produced in VQ. Using this residual codebook the distortion in the reconstructed image is removed, thereby increasing the image quality. Our scheme combines these two methods. Experimental results on standard image Lena show that our scheme can give a reconstructed image with a PSNR value of 31.6 db at 0.396 bits per pixel. Our scheme is also faster than the existing VQ variants.

***Keywords*—**Image compression, Vector Quantization, Residual Codebook.

I. Introduction

VECTOR Quantization [1] [2] has been observed as an efficient technique for image compression. VQ compression system contains two components: VQ encoder and decoder as shown in Fig.1. In VQ method, the given image is partitioned into a set of non-overlapping image blocks $X = \{x_0, x_1, ..., x_{m-1}\}$ of size 4 x 4 pixels each and a clustering algorithm, for example LBG [3], is used to generate a codebook $C = \{Y_0, Y_1, ..., Y_{N-1}\}$ for the given set of image blocks. The codebook C consists of a set of representative image blocks called codewords. The VQ encoder finds a closest match codeword in the codebook for each image block and the index of the codeword is transmitted to VQ decoder. In the decoding phase, VQ decoder replaces the index values with the respective codewords from the codebook and produces the quantized image, called as reconstructed image. In order to achieve low bit rate, many VQ schemes, such as side-match VQ (SMVQ) [4], classified SMVQ (CSMVQ)[5] and Gradient based SMVQ (GSMVQ)[6], have been proposed. SMVQ [4] makes use of the high correlation existing between neighboring blocks to achieve low bit rate. It uses master codebook C to encode the image blocks in the first column and first row in advance.

Manuscript received March 31, 2006.

K.Somasundaram and S.Domnic are with Department of Computer Science & pplications, Gandhigram Rural Institute, Gandhigram-624302, and India (Corresponding author phone: 91-451-2452371; fax: 91-451-2453071, e-mail: somasundaramk@yahoo.com and S.Domnic e-mail: to_domnic@yahoo.co.in).

The other image blocks are encoded utilizing the correlation with the neighboring encoded image blocks. Let x be the input image block, and u and l be the upper and left neighboring codewords respectively. Let the size of the given image block size be $k = m \times n$. The side-match distortion of a codeword Y can be defined as:

$$smd(Y) = \sum_{i=0}^{n-1} (u_{(m-1,i)} - Y_{(0,i)})^2 + \sum_{i=0}^{m-1} (l_{(i,n-1)} - Y_{(i,0)})^2 \quad (1)$$

SMVQ sorts the codewords according to their side-match distortions of all codewords and then selects N_S codewords with smallest side-match distortions from the master book C of size N to form the state codebook SC, where $N_S < N$. A best-match codeword Y_i is selected to encode an image block x from N_S codewords and the corresponding index is coded in $log_2 N_S$ bits. Thus, the SMVQ reduces the bit rate of VQ. Since mean square error caused by state codebook is higher than that of master codebook, SMVQ degrades the image quality and also it requires long encoding time. Classified side-match vector quantization [5] (CSMVQ) is an efficient low bit rate image compression technique which produces relatively high quality image. It is a variable rate SMVQ and makes use of variable sized state codebooks to encode the current image block. The size of the state codebook is decided based on the variances of left codewords and upper codewords that predict the block activity of the input blocks. Also, CSMVQ uses two master codebooks, one for low detail blocks and another for high detail blocks. Another variant, gradient-based SMVQ [6] (GSMVQ) has been proposed, in which gradient values are used instead of variance values to predict the input vector. Another low bit rate VQ, called Jigsaw-puzzle vector quantization (JPVQ) [7] was proposed, in which an input block can be coded by the super codebook, the dynamic codebook or the jigsaw-puzzle block. The jigsaw-puzzle block is constructed dynamically using four-step side-match prediction technique.

However, the low bit rate schemes, SMVQ, CSMVQ, GSMVQ and JSPVQ require high encoding time than that of VQ method. In this paper, we propose an efficient low bit rate image compression scheme based on VQ that makes use of compression of indices of VQ and residual codebook. This scheme achieves low bit rate and better image quality than SMVQ, CSMVQ, GSMVQ and JPVQ.

The rest of the paper is organized as follows: in section II, our compression scheme is described. Performance of our

scheme along with other existing schemes is discussed in section III and the conclusion is given in section IV.

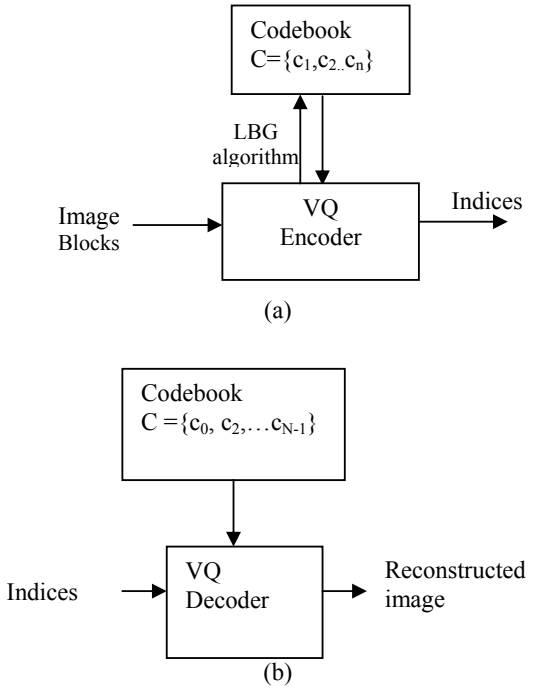

Fig. 1 (a) VQ Encoder (b) VQ Decoder

II. Our Compression Scheme

Our image compressor consists of two components, compression of indices and generation of residual codebook. These two are explained in this section.

A. Compression of Indices

When the image blocks are vector quantized, there likely to exist high correlation among the neighboring blocks and hence among the corresponding codeword indices. Therefore, if indices are coded by comparing with the previous indices, further reduction in the bit rate can be achieved. In Search Order Coding (SOC) [8], [9], a simple searching scheme is followed to find a match for the current index from the previous indices. The search order SO is defined as the order in which the current index is compared with the previous indices. The SO used in [8] and [9] is given in Fig.2 (a) and (b) respectively. The label "1" indicates the highest searching priority, "2" denotes the second highest priority and so on. In order to limit the comparisons of current index with previous indices, the searching range (SR) is fixed. The SR is defined as the number of previous indices to be compared with current index. The previous work [8] has shown that SR = 3 gives the better bit rate than other high SR values. In the Modified Search Order [9], SR is taken as 10, which gives the lower bit rate than other SR.

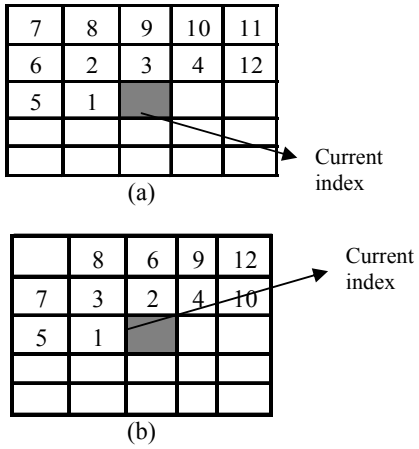

Fig. 2 (a) Searching Order [8] (b) Modified Searching Order [9]

In our scheme, the index of the codeword of a block is encoded exploiting the degree of the similarity of the block with previously encoded upper or left blocks. When the degree of similarity of the current block with one of the two previously encoded blocks is high, the index of the codeword of the block is encoded using the index of the neighboring codeword. I.e. the codeword index of the current block and that of the neighboring blocks are same. If the degree of similarity of the block with the neighboring blocks is not high, we assume that the closest match codeword of the current block may be nearer to the codewords of the neighboring blocks. For example, if one of the two neighboring blocks codeword's index is '*N*', the closest match codeword of the block to be encoded may lie between $(N-J)_{th}$ codeword and $(N+J)_{th}$ codeword in the codebook, where J is any arbitrary number. So the index can be coded in $\log_2(2*J)$ bits. This idea is based on the property existing in the codebook design using LBG algorithm with splitting technique. In the splitting technique, bigger size codebook is generated by splitting each codeword of the smaller codebook into two. The size of the codebook is always in powers of two ($2^M \rightarrow 2^{(M+1)}$). Hence, relatively similar two image blocks may have same closest match codeword in J^{th} position at codebook of size 2^M and at codebook of size $2^{(M+1)}$, one of the two image blocks may have its closest match codeword at J^{th} place in the codebook and other block's codeword may be in $(J+1)^{th}$ place. The other non-similar blocks are encoded using their original index value. In our scheme, examining the resemblance of a block with its left and upper blocks is not required to encode the index of the block. The above description is the idea behind our VQ indices compression scheme. In order to implement our idea, the index to be coded is compared with previous indices according to the SO given in Fig.2.b and SR is fixed as 2 in our scheme. Let 1, 2,..,12 be the SO and ind_val (1), ind_val (2),..ind_val(12) be the indices values of the SO = 1, 2,…12. We use the following steps to encode VQ index.

1 Get the first index generated by the VQ encoder and transmit as such.
2. Get the next index generated by VQ Encoder. Compare this index with the previous indices according SO
3. if SO = 1, code it as "00" and go to the step 2

else
if SO = 2, code it as "01" and go to the step 2
else go to the next step.

4 if index value ≤ (ind_val (SO = 1) + J) and
index value ≥ − (ind_val (SO = 1)+J)
{ if ind_val (SO =1) = ind_val (SO=2)
code it as "10" followed by $\log_2(2*J)$ bits
else
code it as "100" followed by $\log_2(2*J)$
bits. }
go to step 2.
else
if index value ≤ (ind_val (SO = 2) + J) and
index value ≥ − (ind_val (SO = 2)+J)
code it as "101" followed by $\log_2(2*J)$ bits
and go to step 2.
else
code it as "11" followed by its original index and goto step 2.

Decoding of the compressed indces is done by reversing the above coding steps. The performance of our proposed scheme is evaluated with the existing techniques SOC [9] and SOC [10] for different gray- scale images of size 512x512 and is given in the table I. J is set to 4 for our scheme. From table I, we note that our scheme has an improvement in coding the VQ indices.

TABLE I
PERFORMANCE OF OUR METHOD, SOC[8], SOC[9] AND VQ WITH CODEBOOK SIZE 64 IN CODING STANDARD GRAY SCALE IMAGES OF SIZE 512 X 512 EACH

Images	VQ bits/index	SOC[8] bits/index	SOC[9] bits/index	Our method bits/index
Lena	6	4.11	3.92	3.88
Barbara	6	4.29	4.07	4.00
Jet	6	4.22	4.08	3.99
Peppers	6	3.85	3.72	3.66
Zelda	6	4.64	4.45	4.32

B. Construction of Residual Codebook (RC)

Residual codebook $RC = \{RY_0, RY_1, ..., RY_{L-1}\}$ is constructed using absolute error values caused by VQ method, In the residual codebook construction, the image blocks that are less similar to their closest match codewords found in the codebook are taken into account. Less similarity blocks will increase distortion than high similarity blocks in the reconstructed image. Residual codeword (RY_i) for a less similarity image block is constructed by comparing it with its closest match codeword. The collection of residual codewords RY_i, RY_{i+1}… is called residual codebook. Similarity of an image block x with its closet match codeword Y_i is determined based on minimum distortion rule (α) between them. If the mean square error (α) of an image block is greater than a predefined threshold value (σ), then the block is taken as less similarity block. Let $x = (x_0, x_1, .. x_{k-1})$ be a k-pixels image block and $Y_t = \{y_0, y_1, .. y_k\}$ be a k-pixels closest match codeword, then the α is defined as:

$$\alpha = \frac{1}{k}\sum_{i=0}^{k-1}(x_i - y_i)^2 \quad (2)$$

The steps used for constructing residual codebook are given below.

Step1: An image to be compressed is decomposed into a set of non-overlapped image blocks of 4x4 pixels.

Step 2: A codebook is generated for the image blocks using LBG algorithm.

Step 3: Pick up the next codeword Y_t from the codebook C and find its all closest match less similarity image blocks(X) found out using (2) from the given set of image blocks and construct residual codeword RY_t using the following equation.

$$RY_t = \frac{1}{m}\sum_{i=1}^{m}\{|Y_{t1} - X_{i1}|, |Y_{t2} - X_{i2}|, ..., |Y_{tk} - X_{ik}|\} \quad (3)$$

where k denotes the number of elements in the codeword Y_t and the image block X_i respectively and m denotes the number of less similarity image blocks that are closer to the codeword Y_t.

Repeat the step 3 until no more codeword exists in the codebook.

Since residual codeword RY_i is constructed only for less similarity image blocks, some of the codewords Y_i may not have their respective residual codewords, i.e; these codewords may not have less similarity image blocks. In residual codebook construction, only absolute values of the residuals of the less similarity image blocks are used. The sign information for each less similarity image block is preserved and is called residual sign bit plane. In encoding phase, for each less similarity image block, pixels of the block are subtracted from the corresponding pixel values of the codeword Y_i, then sign values (positive or negative) of the residual values of that block, called residual sign bit plane, are preserved. To reduce the bits needed for residual sign bit plane, only alternate bits are stored and others are dropped based on the assumption that there exists correlation among neighboring bits. The bits used for prediction is shown in Fig.3. In the decoding process, the bits of the residual sign bit plane of a block are replaced with the respective residual values of the residual codeword from the residual codebook (RC) with appropriate sign. The residual values of the dropped bits are predicted from neighboring residual values using following steps.

1. $pv(B) = \frac{rrv(A) + rrv(C) + rrv(F)}{3}$
2. $pv(D) = \frac{rrv(C) + rrv(H)}{2}$
3. $pv(E) = \frac{rrv(A) + rrv(I) + rrv(F)}{3}$

4. $pv(G) = \frac{rrv(H) + rrv(C) + rrv(F) + rrv(K)}{4}$
5. $pv(J) = \frac{rrv(I) + rrv(N) + rrv(F) + rrv(K)}{4}$
6. $pv(L) = \frac{rrv(H) + rrv(K) + rrv(P)}{3}$
7. $pv(M) = \frac{rrv(I) + rrv(N)}{2}$
8. $pv(O) = \frac{rrv(N) + rrv(K) + rrv(P)}{3}$

where *pv* (*) is the predicted value of the corresponding bit in the residual sign bit plane and *rrv* (*) is the respective reconstructed residual value of the bit in the residual sign bit plane. After reconstructing the residual codeword, each value of the residual codeword is added to respective value of the closest match codeword of the block

Since the residual sign bit plane for each image block has only eight bits, alternate residual values in the residual codeword RY_t are dropped and it also reduces the cost of storing residual codebook. The dropped residual values are predicted from the neighboring residual values as given above.

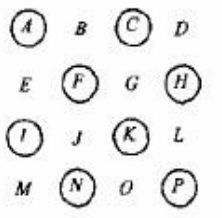

Fig. 3 Bits encircled are used for prediction

C. The Proposed Algorithm

Our scheme combines compression of VQ indices and residual codebook. The steps used in our compressor are.

1. An image to be compressed is decomposed into a set of non-overlapped image blocks of size 4x4 pixels.

2. A codebook is generated for the image blocks using LBG [3] algorithm.

3. Construct a Residual Codebook (as described in section II.B) for those image blocks (less similarity blocks) whose α is greater than σ.

4. Pick the next image block (current block) and find its closest match codeword in the codebook. Calculate mean square error α for the image block using equation (2) and index of the codeword is encoded using VQ indices compression scheme presented in section II.A.

5. if ($\alpha \leq \sigma$), the current block is encoded as "0".
   else
   the current block is encoded as "1" followed by interpolated residual sign bitplane which is computed as described in section II.B.

6. Repeat the step 4 until no more blocks exist in the image.

The decoding of the compressed images is done by reversing the above said steps and residual block to be added is reconstructed for each less similarity block as described in section II.B.

III. Experimental Results and Discussion

To evaluate our scheme we carried out experiments on standard gray scale images using a Pentium-IV computer running at 1.60 GHz under Linux Fedora core-2. For comparison, we also applied other methods CSMVQ, GSMVQ and JPVQ on the same standard images. Three images of 512 x 512 pixels in size are used. Codebook is generated using LBG [3] algorithm for all the methods. For our scheme, a codebook of size 64 is used. For GSMVQ and JPVQ, the codebook size is 256. Performances of the above algorithms are evaluated in terms of bit rate (bits per pixel) and peak signal-to-noise ratio (*PSNR*) given by:

$$PSNR = 10 \log_{10} \frac{(255)^2}{MSE} db \quad (4)$$

where *MSE* (mean squared error) is defined as:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i) \quad (5)$$

where x_i and y_i denote the original and the encoded pixel values and *n* is the total number of pixels in an image. Bit rate including overhead bits (i.e bits need to store codebook) for different threshold values ranging from 50 to 2000 for Lena Jet and Pepper are given in Figs. 4, 5 and 6. From Figs. 4, 5 and 6, we observe that our scheme gives PSNR values of 31.60db, 31.820db and 32.340 db at the bit rates of 0.396bpp, 0.410bpp and 0.479bpp respectively for Lena image. The other methods, JPVQ gives PSNR values of 31.460db, 31.710db and 31.830db at the bit rates of 4.00bpp, 0.438bpp and 0.538bpp respectively, GSMVQ gives PSNR values of 31.150db, 31.350db and 31.680 at the bit rates of 0.400bpp, 0.425bpp and 0.459bpp for the same image. Since our scheme uses smaller codebook in VQ method, it gives less encoding time than GSMVQ and JPVQ.

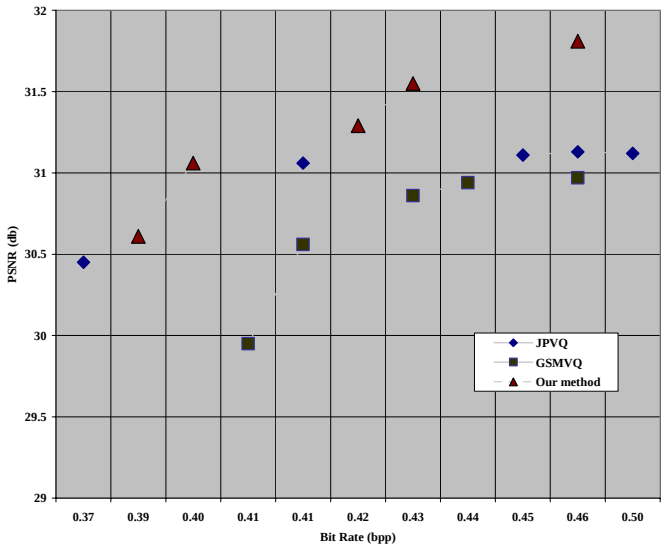

Fig. 4 Experimental results in terms of PSNR and bit rate for JPVQ, GSMVQ and our method with Lena image as the standard test image

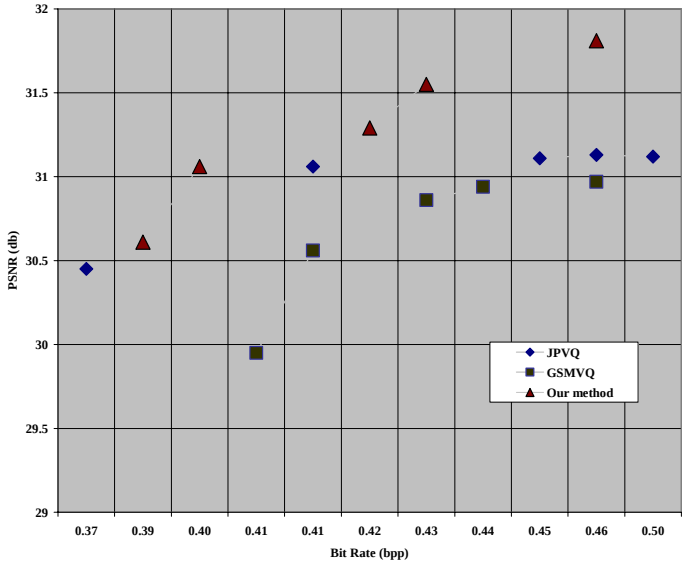

Fig. 5 Experimental results in terms of PSNR and bit rate for JPVQ, GSMVQ and our method with Jet as the standard test image

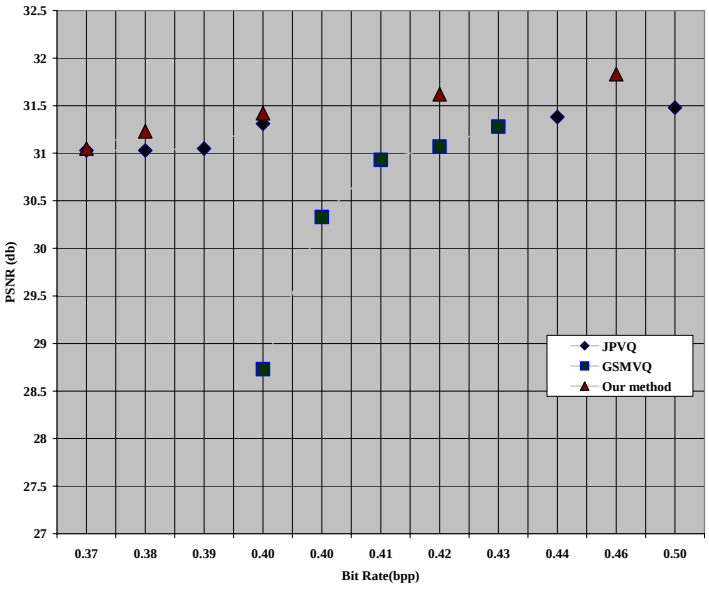

Fig. 6 Experimental results in terms of PSNR and bit rate for JPVQ, GSMVQ and our method with Pepper as the standard test image

V. Conclusion

In this paper, we have proposed a new gray scale image compression scheme which gives better image quality and low bit rate. This scheme is based on VQ method and employs residual codebook to improve image quality and compression of VQ indices to lower the bit rate. Experimental results on standard images show that our scheme gives better PSNR values and low than GSMVQ and JPVQ. Since our scheme uses smaller codebook, it gives faster compression than the other two schemes.

References

[1] R. M. Gray, "Vector quantization," IEEE Acoustics, speech and Signal Processing Magazine, pp. 4-29, 1984.
[2] M. Goldberg, P. R. Boucher and S. Shlien, "Image Compression using adaptive vector quantization," IEEE Transactions on Communication, Vol. 34, No. 2, pp. 180-187, 1986.
[3] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design," IEEE Transactions on Communication, Vol. 28, No. 1, 1980, pp. 84 – 95.
[4] T.Kim, "Side match and overlap match vector quantizers for images," IEEE Trans. Image. Process., vol.28 (1), pp.84-95, 1980.
[5] Z.M.Lu, J.S Pan and S.H Sun, "Image Coding Based on classified side-match vector quantization," IEICE Trans.Inf.&Sys., vol.E83-D(12), pp.2189-2192, Dec. 2000.
[6] Z.M.Lu, B.Yang and S.H Sun, "Image Compression Algorithms based on side-match vector quantizer with Gradient-Based classifiers," IEICE Trans.Inf.&Sys., vol.E85-D(9), pp.1414- 1420, September. 2002.
[7] Chia-Hung Yeh, "Jigsaw-puzzle vector quantization for image compression" , Opt.Eng Vol.43, No.2, pp. 363-370, Feb-2004.
[8] C.H.Hsieh, and J.C Tsai, "Lossless compression of VQ index with search order Coding," IEEE Trans. Image Processing, Vol.5, No. 11, pp. 1579- 1582, Nov. 1996.
[9] Chun-Yang Ho, Chaur-Heh Hsieh and Chung-Woei Chao, "Modified Search Order Coding for Vector Quantization Indexes," Tamkang Journal of Science and Engineering, Vol.2, No.3, pp. 143- 148, 1999.