IMAGE COMPRESSION USING VECTOR QUANTIZATION

Nopparat Pantsaena[1], M. Sangworasil[1], C. Nantajiwakornchai[1]and T. Phanprasit[2]
[1]The Research Center for Communication and Information Technology (ReCCIT)
Dept. of Electronics, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Chalongkrung Road., Ladkrabang, Bangkok 10520, Thailand
Tel: +66-2-326-9968
e-mail: ksamanas@kmitl.ac.th
[2]Department of Electronics Engineering
School of Engineering
Bangkok University
Pathumtani, Bangkok 12120, Thailand
Tel: +66-2-902-0299 Ext. 620,650
e-mail: tanasak.p@bu.ac.th

Abstract: Compressing image data by using Vector Quantization (VQ)[1]-[3] will compare Training Vectors with Codebook. The result is an index of position with minimum distortion. The implementing Random Codebook will reduce the image quality. This research presents the Splitting solution [4],[5] to implement the Codebook, which improves the image quality[6] by the average Training Vectors, then splits the average result to Codebook that has minimum distortion. The result from this presentation will give the better quality of the image than using Random Codebook.

1. Introduction

From source images (512x512), divide the image into small blocks (4x4). There are 16,384 blocks that will be reshaped as Training Vectors of 16,384x16 (Table 1.) and Codebook of 256x16 (Table 2.). There are two kinds of Codebook: Random Codebook (256x16) random sample from Training Vectors, and Split Codebook (256x16) average value of each column of Training Vectors. Split Codebook is 1x16. Then, follow the Vector Quantization process.

In the Image data compression process VQ type as shown in figure 1, Codebook is an essential part of the image quality, because the image quality is the result of the comparison between Training Vectors with Codebook. This paper will present the two types of Codebook which are used in the compression process. They are Random Codebook and Split Codebook. Split Codebook is obtained from the average of the Training Vectors, and then split the average into 2 vectors. And then one.vector is added by the average value from itself. While the other one in subtracted as shown in figure 2, observe the average distortion which is the result of the comparison between the Training Vectors and each of the two vectors whether it is equal, less than, or more than the decision value. If it is more than the decision value, each vector will be split into two. Otherwise, stop, and follow the VQ process. Split Codebook is the average amount of Training Vectors population. The result is reducing scatter data better than random sample (Random Codebook).

2. Vector Quantization

In Image Compression using Vector Quantization, an input image is divided into small blocks called Training Vectors ($x_j(k)$). This Training Vectors can be closely reconstructed from applying a transfer function (Q) to a specific region of an input image itself, which is called Codebook ($\hat{x}_i(k)$). Thus, only the set of transfer functions, which have fewer data than an image, were required for reconstruct the input image back. A transfer function (Q) is defined as follows.

$$Q : R^k \to Y \tag{1}$$

That Y= $\{\hat{x}_i\ ; i = 1,2,3,\ldots, N\}$ is the set of vectors

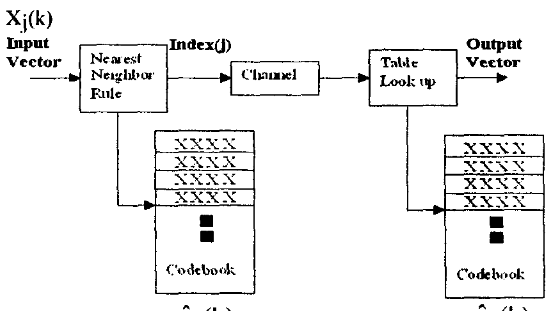


Figure 1. Block diagram of Vector Quantization.

that recreate and N is the number of vectors in Y. VQ consists of 2 parts. First encoder will compare the Training Vector and Codebook Vectors. The result is an index (j) that represents the position of Codebook with minimum distortion. Second, decoder will bring the index to direct the same Codebook as encoded.

Codebook design

Searching for the best Codebook from studying the training set, Linde-Buzo-Gray's Algorithm (LBG) is the famous algorithm for Codebook. LBG starts with define in the value of Y_m, then update Codebook by replacing the value that makes less distortion. This update step will iterate until the distortion is under the LBG limit. LBG instructions will split the image into the same size parts and forms to Training Vectors $x_j(k)$ as: $x_j(k)$=[$x_j(1)\, x_j(2)\, x_j(3)\ldots x_j(k)$]

j = 1,2,3,...,n; n = number of the image parts
k = vector dimension; (k=1,2,3,...,16)

Steps of LBG

1. Initiate the value for Codebook $\hat{x}_i(k)=[\hat{x}_i(1)\,\hat{x}_i(2)\,\hat{x}_i(3)\ldots\hat{x}_i(k)]$; i=1,2,3,...,N$_c$ and define the decision value (ε); N$_c$ is number of vector in Codebook
2. Vector group(X_j) (Table 3) is the Training Vectors by the number of Codebook Vectors P($\hat{x}_i(k)$) = (S_i; i=1,2,3,...,Nc); S_i is the set of minimum distortion vectors The grouping will use the minimum distortion values, which can find from: $x_j \in S_i$ if $d(x_j,\hat{x}_i) \le d(x_j,\hat{x}_l)$ for every i we can obtain the minimum distortion values from:

$$d(x_j,\hat{x}_i) = \frac{1}{k}\sum_{m=1}^{k}[x_j(m)-\hat{x}_i(m)]^2 \qquad (2)$$

3. Average distortion (D_{m+1})
The total number of group, which are computed, between Xj(k) and Xi(k) is 256 groups. Then the average distortion can be calculated from each group, is given by

$$D_{m+1} = D[(Ym, P(Ym))] = \frac{1}{n}\sum_{j=1}^{n}\min_{\hat{x}\in Ym} d(x_j,\hat{x}_i) \qquad (3)$$

4. If $\frac{D_m - D_{m+1}}{D_{m+1}} \le \varepsilon$, stop update the Codebook. we can obtain the New Codebook ($\hat{x}_i(k)$' (Table 4). Otherwise, the process will continue update in the Codebook by average the group in 2nd step vectors as: $\hat{x}(P(Ym)) = (\hat{x}(S_i)\,;\, i = 1,2,3,\ldots, Nc)$ for P($\hat{x}_i(k)$)

where $$\hat{x}(S_i) = \frac{1}{\|S_i\|}\sum_{j:x_j\in S_i}^{m} x_j \qquad (4)$$

The result $Y_{m+1} = \hat{x}(S_i)$ increase i by 1, and then back to the 1st step also the Codebook design presentation in table 1-4.

The decoding of the New Codebook and the suitable index yields the output vector. That becomes data storage in Training Vector and Codebook Vectors region according to the image compression .

Table 1. Training Vector

Index (j)	Training Vector ($x_j(k)$)
1	$x_1(1)\ldots x_1(16)$
2	$x_2(1)\ldots x_2(16)$
3	$x_3(1)\ldots x_3(16)$
M	M
16,384	$x_{16384}(1)\ldots x_{16384}(16)$

Table 2. Codebook

Index (i)	Codebook ($\hat{x}_i(k)$)
1	$\hat{x}_1(1)\ldots\hat{x}_1(16)$
2	$\hat{x}_2(1)\ldots\hat{x}_2(16)$.
3	$\hat{x}_3(1)\ldots\hat{x}_3(16)$
M	M
256	$\hat{x}_{256}(1)\ldots\hat{x}_{256}(16)$

Table 3. Vector group

Index (i)	Group 1 ($X_{j=1}(k)$)
1	$X_{1,1}(1)\ldots X_{1,1}(16)$
2	$X_{1,2}(1)\ldots X_{1,2}(16)$

M

$\mathrm{I}_1 \in X_{j=1}$

Index (i)	Group 2 ($X_2(k)$)
1	$X_{2,1}(1)\ldots X_{2,1}(16)$
2	$X_{2,2}(1)\ldots X_{2,2}(16)$

M

$\mathrm{I}_2 \in X_{j=2}$

Index (i)	Group 3 ($X_3(k)$)
1	$X_{3,1}(1)\ldots X_{3,1}(16)$
2	$X_{3,2}(1)\ldots X_{3,2}(16)$

M

$\mathrm{I}_3 \in X_{j=3}$

Index (i)	Group 256 ($X_{256}(k)$)
1	$X_{256,1}(1)\ldots X_{256,1}(16)$
2	$X_{256,2}(1)\ldots X_{256,2}(16)$

M

$\mathrm{I}_{256} \in X_{j=256}$

Table 4. New Codebook

Index (i)	New Codebook ($\hat{x}_i(k)$')
1	$\hat{x}_1(1)'\ldots\hat{x}_1(16)'$
2	$\hat{x}_2(1)'\ldots\hat{x}_2(16)'$
3	$\hat{x}_3(1)'\ldots\hat{x}_3(16)'$
M	M
256	$\hat{x}_{256}(1)'\ldots\hat{x}_{256}(16)'$

3. Splitting Codebook design

In some case, a product codebook may provide a good initial guess. For example, if one wishes to design a Codebook for a k-dimensional VQ with codebook size 2^{KR} for some integral resolution R, then one can use the product of K scalar quantizers with 2^R words each. Thus if q (x) is a scalar quantizer, then Q ($x_0,\ldots,x_{k-1}$) = (q (x_0),...,q(x_{k-1})), the Cartesian product of the scalar quantizer, is a Vector Quantizer. This technique will not work if R is not an integer. But we can use the Spitting algorithm; it is not an integral number of bits per symbol [7]. Of course, any *practical* VQ codebook must be designed from a suitable training set as shown in Table 1-4.

From Training Vector, find the average of Training Vectors for setting the Codebook as follows.

$$\hat{x}_1(k) = \frac{1}{n}\sum_{j=1}^{n} x_j(k) \qquad (5)$$

k=1,2,3,...,16
n =1,2,3,...,1638

Split the Codebook to 2 vectors: $\hat{x}_1(1)+\varepsilon_1$ and $\hat{x}_1(1)-\varepsilon_1$, when ε_1 is the average value as shown in figure 2.

$$\varepsilon_j = \frac{\sqrt{\sum_{i=1}^{n_i}(\hat{x}_i - \mu_j)^2}}{n_i} \tag{6}$$

$n_i = 1,2,3,\ldots,16$

$$\mu_j = \frac{1}{n_i}\sum_{i=1}^{n_i}\hat{x}_{ij} \tag{7}$$

$j = 1,2,3,\ldots,Nc$

Find the New Codebook ($\hat{x}_i(k)'$) , then take LBG give minimal error. (in 2nd step)

Y_m

$\hat{x}_1(1)$	$\hat{x}_1(2)$	$\hat{x}_1(3)$	…	$\hat{x}_1(16)$

=

Y_{2m}

$\hat{x}_1(1)+\varepsilon_1$	$\hat{x}_1(2)+\varepsilon_1$	$\hat{x}_1(3)+\varepsilon_1$	…	$\hat{x}_1(16)+\varepsilon_1$

$\hat{x}_1(1)-\varepsilon_1$	$\hat{x}_1(2)-\varepsilon_1$	$\hat{x}_1(3)-\varepsilon_1$	…	$\hat{x}_1(16)-\varepsilon_1$

Fig 2. Splitting the Codebook to 2 vectors

4. Experimental & result

This experimentation shows the image encoding by using 4 images ("Lena, Gird, Milkdrop, Mandrill",8 bit/pixel, 512*512 pixel) in the process, including comparing the image quality from Random and Splitting Codebook are shown in Fig. 3. By this comparison, the presented quality is better than Random Codebook type for all the test images. And the value of PSNR is calculated from

$$PSNR = 20\log_{10}\left(\frac{A}{RMSE}\right) \tag{8}$$

That A is the maximum of gray level and RMSE is Root Mean Square between the original image and the decompressed image.

$$\left[RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{m}[f(i,j) - f(i,j)']^2} \right] \tag{9}$$

when f(i,j) is the gray level of pixel i in the original image
f(i,j)' is the gray level of pixel i in the decompressed image.
N is the total number of pixels in the image.

In the comparison between Random Codebook and Split Codebook to find the better initial Codebook for image compression, the result reveals that Split Codebook produces better image quality.

For "Milkdrop ", which has a lot of low frequency components, the image quality is a little better at low bits rate.

For "Mandrill", which has a large amount of high frequency components, the image quality is a little better at low bits rate. Except at high bits rate that our method is not sufficient.

For "Girl", the result is similar with "Lena".

For "Lena", the image quality is a little better at all bits rate.

5. Conclusion

This paper proposed the implementaion of Codebook to Vector Quantization image compression. From the experiment, the Split Codebook can achieve higher image quality than Random Codebook at the same bits rate. In another words, the proposed algorithm achieves lower bits rate at the same image quality.

References

[1] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Communs.*, vol. COM-28, pp. 84-95, 1980.

[2] C. H. Lee, "Study on Vector Quantization for Image Compression," *Ph.D. dissertation, Institute of Computer and Information,* NCTU, Hsinchu, Taiwan, 1995.

[3] N. M. Nasrabadi and R.A. King, "Image coding using vector quantization: A Review," *IEEE Trans. Commun.,* vol. COM-36, pp. 957-971, Aug 1980.

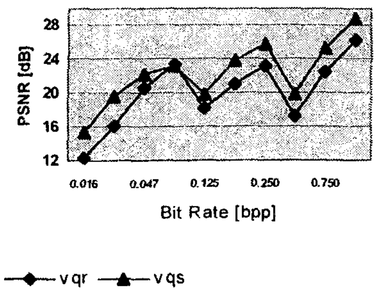
[4] Peter Vprek, and A. B. Bradley, " An Improved Algorithm for Vector Quantizer Design," *IEEE Signal processing letters,* vol. 7, No.9, pp.250-252, September 2000.

[5] Kai Bao and Xiang-Gen Xia, "Image Compression Using a New Discrete Multiwavelet Transform and a New Embedded Vector Quantization ," *IEEE Trans. On circuits and systems for video technology,* vol 10, No. 6, pp. 833-842, September 2000.

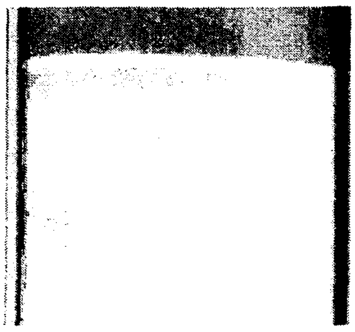
[6] S. Wongkhran "Satellite imagery data compression using Vector Quantization," *Thesis submitted, Electrical Engineering,* King Mongkut's Institue of Technology Ladkrabang, 1997.

[7] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. Comm.,* COM-28:84-95, January 1980.

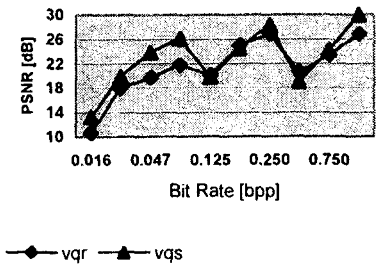
(a)



(b)

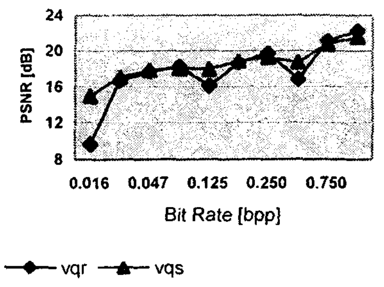


(c)

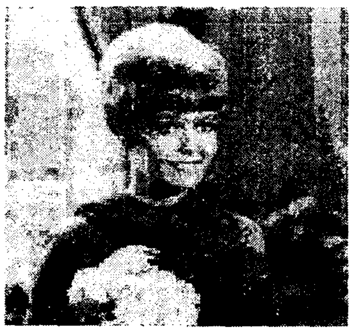


(d)

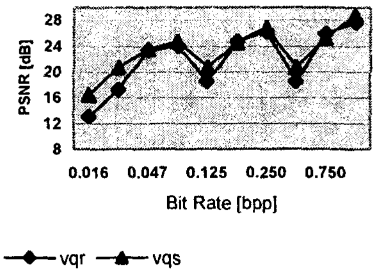
(e)



(f)



(g)



(h)

Figure 3. The decoded image of Split Codebook (a) "Lena" (22.087 [dB], 0.047 [Bpp]), (c) "Milk drop" (23.819 [dB], 0.047 [Bpp]), (e) "Mandrill" (17.795 [dB], 0.047 [Bpp]), (g) "Girl" (23.455 [dB], 0.047 [Bpp]) ; (b), (d), (f) and (h) show the graph of PSNR versus Bit rate for each image.