Codebook Generation for Vector Quantization using Interpolations to Compress Gray Scale Images

S.Vimala
Dept. of Comp. Sci.
Mother Teresa Women's University
Kodaikanal – 624 102

K.KowsalyaDevi
Dept. of Comp. Sci.
Mother Teresa Women's University
Kodaikanal – 624 102

M.Sathya
Dept. of Comp. Sci.
Mother Teresa Women's University
Kodaikanal – 624 102

ABSTRACT
Vector Quantization (VQ) is one of the Lossy Image Compression Techniques. Vector Quantization is performed in three phases: 1.CodeBook Generation, 2.Image Encoding and 3.Image Decoding. In this paper, we have proposed a novel idea of incorporating the technique of Interpolation as part of the existing codebook generation technique to further reduce the bit-rate that is obtained. Experiments were carried over standard images like Lena, Cameraman, Boats, Bridge, Baboon, Goldhill, and Kush. According to the simulated results, the proposed algorithm achieves significant reduction in bit-rate for codebooks of various sizes such as 16, 32, 64, 128, 256 and 512.

General Terms
Image Processing, Compression, Vector Quantization.

Keywords
vector quantization, codebook, encoding, decoding, bit-rate, interpolation, image compression

1. INTRODUCTION
The image compression [1] techniques are broadly classified into two categories: 1.Lossless techniques and 2.Lossy techniques. In lossless compression techniques, the reconstructed image will be the exact replica of the original image. Some of the lossless compression techniques: 1.Run Length Encoding, 2.Huffman Encoding, 3.LZW coding and 4.Area coding [2]. In lossy compression techniques, the decompressed image is not identical to the original image, but an approximation of the original image [3]. Some of the lossy compression techniques: 1.Transformation Coding, 2.Vector Quantization, 3.Fractal Coding, 4.Block Truncation Coding and 5. Subband Coding [4].

Vector Quantization (VQ) is a classical quantization technique from signal processing and image compression [5] which allows the modeling of probability density functions by the distribution of prototype vectors. VQ assists to project a continuous input space on a discrete output space, while minimizing the loss of information [6]. The purpose of VQ [7] originally stems from encoding discrete data vectors to compress data which have to be transferred quickly e.g for online communication channels. The performance of the VQ is known to be asymptotically optimum in the sense of minimum rate distortion as the vector dimension increases [8], [9], [10].

VQ has been widely used for signal compression due to its excellent rate-distortion performance. The excellent review on speech and image applications can be found in [11] [12].

VQ is done in three steps: 1.Codebook Design, 2.Image Encoding and 3.Image Decoding.

The performance of the VQ depends mainly on the quality of the codebook that is generated. There are several known methods for generating codebook. The most cited and widely used is the Linde Buzo Gray (LBG) [13] Algorithm. It starts with an initial solution, which is iteratively improved using two optimality criteria until a local minimum is reached [14]. Another hierarchical algorithm, the Pairwise Nearest Neighbor (PNN) [15] uses an opposite, bottom-up approach to generate the codebook.

In codebook generation, an image is split up into blocks of size 4 x 4 pixels. The blocks are converted into vectors of dimension K. These vectors are called training vectors, and the set of training vectors is called the training set of size N vectors [16]. N is computed using the equation (1).

$$N = (m \times m) / k \quad (1)$$

where k is the dimension of the vector and m x m is the number of rows and columns of pixels in the input image. The position p from which the codevectors to be selected is computed using the equation (2).

$$p = N / M \quad (2)$$

where N is the size of the Training Set and M is the desired size of the codebook. The codevectors at every p^{th} position from the training set are selected to form the codebook.

In encoding stage, the closest codevector for each training vector is identified by computing the distance d(Y, Xi) using the equation (3).

$$d(Y, X_i) = \sum_{j=1}^{16} (y_j - x_{ij}) \quad (3)$$

where, i =1 to M and d(Y, Xi) must be minimum for i ≠j. Each block of the input image is replaced with the index of the codevector satisfying equation (3). The collection of such indices is called the Index Map. Now the compressed image is either stored or transmitted in the form of Index Map and the Codebook.

In the decoding stage, while reconstructing the image from the compressed image, each index in index map is replaced with the corresponding codevector. The difference between the input image and the reconstructed image is the Mean Square Error (MSE) and is computed using the equation (4). The Peak Signal to Noise Ratio (PSNR) is the quality of the reconstructed image and is computed using the equation (5).

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (I_i - I_i')^2 \quad (4)$$

$$PSNR = 10 Log_{10} \left(\frac{255^2}{MSE} \right) \quad (5)$$

Our proposed idea is to incorporate the interpolation technique which has been adopted as part of Block Truncation Coding [17] for image compression to further reduce the bit-rate of the compressed image.

The Organization of the paper is as follows: The existing codebook generation method is explained in Section 2, the proposed method is explained in Section 3, the results are discussed in section 4 and the Conclusion is given in Section 5.

2. EXISTING METHODS

2.1 Linde Buzo Gray (LBG) Algorithm

Step1: Computing centroid for the entire training set

Step2: Add a constant error {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1} to the centroid

Step3: Adding a constant error to and subtracting the constant error from the centroid respectively

Step4: Two clusters are formed by grouping the nearest vectors of V1 and V2 using the minimum distance method

Step5: Repeat Step3 until the desired the size of codebook is generated.

2.2 Kekre's Proportionate Error (KPE) Algorithm [18]

Step1: Computing centroid for the entire training set

Step2: Add a proportionate error to the covector. Magnitude of elements of the codevector decides he error ratio.

Step3: Hereafter the procedure is same as that as LBG.

Step4: While adding proportionate error a safe guard is also introduced so that neither V1 nor V2.

Step5: Go beyond the training vector space.

2.3 Simple Codebook Generation (SCG) Algorithm [19]

Step1: Input the given image of size m × m pixels.

Step2: Divide it into blocks of size 4×4 pixels.

Step3: Generate N training vectors using equation (1).

Step4: Fix the codebook size to M.

Step5: Compute p=N/M

Step6: Select every p^{th} training vector as codevector to generate the codebook.

2.4 Interpolation Technique

The Interpolation is adopted in the Block Truncation Coding (BTC) based image compression methods to have a better compression ratio. In this method, the boldfaced components as in Fig. 1 are dropped to improve the coding efficiency.

X1	**X2**	**X3**	X4
X5	X6	X7	**X8**
X9	X10	X11	**X12**
X13	**X14**	**X15**	X16

Fig. 1: Dropping pattern of bits in interpolation technique

In decoding phase, the dropped bits are recovered by taking the arithmetic mean of the adjacent pixel values as given in equation set (6).

$$\left.\begin{aligned} X_i &= \frac{1}{2}(x_{i-1} + x_{i+4}) \text{ for i} = 2,12 \\ X_i &= \frac{1}{2}(x_{i+1} + x_{i+4}) \text{ for i} = 3,9 \\ X_i &= \frac{1}{2}(x_{i-4} + x_{i+1}) \text{ for i} = 5,15 \\ X_i &= \frac{1}{2}(x_{i-1} + x_{i-4}) \text{ for i} = 8,14 \end{aligned}\right\} \quad (6)$$

3. PROPOSED METHOD

3.1 Modified Interpolation Technique

The existing interpolation order is slightly changed and the dropped elements are generated using the equation set (7).

$$\left.\begin{aligned} X_i &= \frac{1}{3}(x_{i-1} + x_{i+4} + x_{i+5}) \text{ for i} = 2 \\ X_i &= \frac{1}{3}(x_{i+1} + x_{i+3} + x_{i+4}) \text{ for i} = 3 \\ X_i &= \frac{1}{3}(x_{i-4} + x_{i+1} + x_{i+5}) \text{ for i} = 5 \\ X_i &= \frac{1}{3}(x_{i+4} + x_{i+1} + x_{i-3}) \text{ for i} = 9 \\ X_i &= \frac{1}{3}(x_{i-4} + x_{i-1} + x_{i+3}) \text{ for i} = 8 \\ X_i &= \frac{1}{3}(x_{i-1} + x_{i-5} + x_{i+4}) \text{ for i} = 12 \\ X_i &= \frac{1}{3}(x_{i-1} + x_{i-4} + x_{i-3}) \text{ for i} = 14 \end{aligned}\right\} \quad (7)$$

3.2 Simple codebook Generation with Interpolation(SCGI)

In the proposed method, both the Simple Codebook Generation technique and the Interpolation technique adopted in BTC are combined and incorporated as part of SCG. The algorithm is as follows:

3.2.1 Encoding Algorithm

1. Generate the Simple Codebook as stated earlier.
2. Generate the Index Map.
3. Drop the elements of all the codevectors of the codebook as in Fig.1.
4. Store or transmit the compressed image in the form of Codebook and Index Map.

3.2.2 Decoding Algorithm

1. Generate the dropped elements of the codevectors using the equation set (7).
2. Reconstruct the image by replacing each index from the Index Map with its corresponding Codevector.

4. RESULTS AND DISCUSSION

Using the proposed (SCGI) technique, codebooks of different sizes 16, 32, 64, 128, 256 and 512 are generated. The images Lena, Cameraman, Boats, Bridge, Baboon, Goldhill and Kush are used for the study and are given in Fig. 2.

a) Lena

b) Cameraman

c) Boats

d) Bridge

e) Baboon

f) Goldhill

g) Kush

Fig 2: Input Images taken for the study

TABLE 1. PSNR AND BITRATE OBTAINED WITH VARIOUS CODEBOOK SIZES

Image	Codebook Size	SCG		SCGI	
		bpp	PSNR	bpp	PSNR
Lena	16	0.28	24.22	**0.26**	**24.30**
	32	0.38	24.82	**0.34**	**24.91**
	64	0.50	25.09	**0.40**	**25.17**
	128	0.68	29.24	**0.56**	29.17
	256	1.00	30.66	**0.75**	30.27
	512	1.56	32.23	**1.06**	31.21
Cameraman	16	0.28	23.96	**0.26**	**23.98**
	32	0.38	24.19	**0.34**	**24.30**
	64	0.50	24.37	**0.40**	24.39
	128	0.68	26.15	**0.56**	25.91
	256	1.00	26.80	**0.75**	26.34
	512	1.56	28.00	**1.06**	26.81
Boats	16	0.28	7.79	**0.26**	7.67
	32	0.38	07.86	**0.34**	**09.73**
	64	0.50	07.86	**0.40**	07.73
	128	0.68	26.40	**0.56**	24.19
	256	1.00	28.33	**0.75**	24.78
	512	1.56	29.56	**1.06**	25.14
Bridge	16	0.28	24.17	**0.26**	23.96
	32	0.38	25.16	**0.34**	**25.18**
	64	0.50	26.23	**0.43**	25.94
	128	0.68	26.02	**0.56**	25.53
	256	1.00	26.90	**0.75**	26.22
	512	1.56	27.98	**1.06**	26.81

TABLE 2. COMPARISON OF THE PSNR OBTAINED WITH LBG, KPE AND PROPOSED METHOD.

Codebook Size	Image	LBG		KPE		Proposed	
		bpp	PSNR	bpp	PSNR	bpp	PSNR
128	Baboon	0.68	21.07	0.68	21.26	**0.56**	**30.91**
	Goldhill	0.68	26.97	0.68	26.98	**0.56**	**33.10**
	Kush	0.68	28.79	0.68	28.80	**0.56**	**29.17**
	Lena	0.68	27.21	0.68	27.27	**0.56**	**29.17**
256	Baboon	1.00	21.33	1.00	21.83	**0.75**	**31.26**
	Goldhill	1.00	27.21	1.00	27.29	**0.75**	**34.09**
	Kush	1.00	29.00	1.00	29.13	**0.75**	**30.00**
	Lena	1.00	27.47	1.00	27.84	**0.75**	**30.27**
512	Baboon	1.56	21.67	1.56	22.72	**1.06**	**31.94**
	Goldhill	1.56	27.62	1.56	28.09	**1.06**	**34.86**
	Kush	1.56	29.39	1.56	29.90	**1.06**	**30.64**
	Lena	1.56	27.93	1.56	28.91	**1.06**	**31.21**

The Results observed for different codebook sizes are listed in the TABLE 1 and TABLE 2. For all images, with the codebook of size 32, the proposed method outperforms both in terms of PSNR and bpp. For Lena image, the bpp and the PSNR obtained with the proposed method (SCGI) are 0.34 and 24.91 respectively, but in SCG, the bpp and PSNR are only 0.38 and 24.82. For codebooks of lowers sizes such as 16 and 32, the proposed method gives better results both in terms of bpp and PSNR. For higher size codebooks, the bpp is reduced significantly with slight degradation in PSNR. For a codebook of size 32, the bpp is reduced from 0.38 to 0.34. For a codebook of size 512, the bpp is reduced from 1.56 to 1.06, which is significant improvement in coding efficiency. In Table 2, the comparison of LBG, KPE and the proposed methods is given with respect to PSNR for codebooks of sizes 128, 256 and 512. In Fig.3, the reconstructed images using the SCG and the SCGI techniques are given.

Original SCG (bpp: 1.00) Proposed (bpp: 0.75)

Original SCG (bpp: 1.00) Proposed (bpp: 0.75)

Fig 3: Reconstructed images of Lena and Cameraman for a codebook of size 256

5. CONCLUSION

In this paper, we have proposed a codebook ceneration technique SCGI for Vector Quantization using Interpolation. This method was used generate codebooks of different sizes 16, 32, 64, 128, 256 and 512. The methods, when tried with standard images, gave better bpp. The PSNR obtained with the proposed method is better when compared to that few existing methods such as LBG and KPE. This method is suitable for handheld devices. The same technique can also be extended for color images.

6. REFERENCES

[1] Sonal, Dinesh,Kumar, "A study of various Image Compression Technique", National Conference on Challenges and Opportunities in Information Technology, Hisar, 2007.

[2] David Solomon, "Data Compression", The Complete Reference, Third Edition.

[3] Somasundaram and S.Vimala, "Codebook generation for vector quantization with Edge Features", CiiT International Journal of Digital Image Processing, 2010.

[4] Rafael C.Gonzalez, Richard E.Woods, "Digital Image Processing", Second Edition.

[5] G.Boopathty and S.Arockiasamy, " Implementation of Vector Quantization for Image Compression – A Survey", Global Journal of Computer Science and Technology, vol.10, Issue 3, pp. 22-28, April 2010.

[6] K.Makhoul, S.Roucos, H.Gish, " Vector Quantization is speech coding", IEEE, vol. 73, n.11, November 2005.

[7] R.Gray, "Vector Quantization", IEEE ASSP Mag., pp. 4-29, 1989.

[8] A.Gersho, "Asymptotically optimal block quantization", IEEE Trans. Inform. Theory, vol. IT-25, pp. 373-380, July 1979.

[9] Y.Yamada, S.Tazaki, and R.M.Gray, "Asymptotic performance of block quantizers with difference distortion measure", IEEE Trans. Inform. Theory, vol. IT-26, pp.6-14, Jan 1980.

[10] T.Berger," Rate Distortion Theory", Englewood Cliffs, NJ: Prentice Hall, 1971.

[11] A.Gersho and R.M.Gray, "Vector Quantization and Signal Compression", Norwell, MA:Kluwer, 1992.

[12] N.M.Nasradbadi nad R.A.King, "Image Coding Using Vector Quantization: A review", IEEE Trans. Commun., vol. 39, pp. 957-971, Aug 1988.

[13] Y.Linde, A.Buzo and R.M.Gray," An Algorithm for Vector Quantizer Design", IEEE Trans. Commun., vol. COM -28, no. 1, pp. 84-95, 1980.

[14] H.B.Kekre, Tanuja K.Sarode, "An Efficient Fast Algorithm to Generate Codebook for Vector Quantization", IEEE, 2008.

[15] William H.Equitz, "A New Vector Quantization Clustering Algorithm", IEEE Trans. On Acoustics, Speech and Signal Processing, vol. 37, No. 10, October 1989.

[16] S.Vimala, "Convergence Analysis of Codebook Generation Techniques for Vector Quantization using k-means Clustering Techniques", International Journal of Computer Applications, vol 21, No.8, May 2011.

[17] S.Vimala, P.Uma Edwin and P.Anne Raja Reega Ruth, "Block Truncation Coding using Enhanced Interpolations and Lookup procedures for Image Compression", International Journal of Computer Applications, vol 29, No.1, Sep 2011.

[18] H.B.Kekre, Sudeep Thepade and Nikita Bhandari, "Colorization of Greyscale Images Using Kekre's Biorthogonal Color Spaces and Kekre's Fast Codebook Generation", International Journal of Advances in Multimedia, vol.1, Issue 3, 2009.

[19] Somasundaram and S.Vimala, "Simple and Fast Ordered Codebook for Vector Quantization", Proceeding of the National Conference on image processing, Gandhigram rural Institute, March 2010.