Journal of Information Hiding and Multimedia Signal Processing ©2018 ISSN 2073-4212
Ubiquitous International Volume **9**, Number **1**, January **2018**

Edge Classification Based Initialization Technique for Image Vector Quantizer Design

Xiao-Dan Jiang

College of Electrical and Information Engineering
Quzhou University
Quzhou, 324000, P. R.China
16282409@qq.com

Zhe-Ming Lu*

School of Aeronautics and Astronautics
Zhejiang University
Hangzhou, 310027, P. R.China
*Corresponding author: zheminglu@zju.edu.cn

Received February 2017;Revised October 2017

Abstract. *Codebook design is one of key techniques in Vector Quantization (VQ). The K-means algorithm is the most widely used algorithm to design a VQ codebook. However, both convergence speed and performance of the generated codebook depend on the initial codebook. Thus, many algorithms have been presented to obtain a good initial codebook, including the well known splitting, pruning, pairwise nearest neighbor design, random initialization and maximum distance initialization. However, all these algorithms do not make full use of the features of the training vectors to obtain a better initial codebook and some of them need high extra overhead. This paper provides a very simple technique to obtain a better initial codebook by classifying training vectors according to their edge orientations dependent on eight edge templates proposed by us, and thus we can obtain eight subsets. Then we randomly select codewords from each subset with the number of codewords being proportional to the number of training vectors in the subset. Experimental results show that, compared with the conventional and modified K-means algorithms based on random selection initialization, our initialization technique converges to a better locally optimal codebook with a faster convergence speed.*
Keywords: Vector quantization, Image compression, Codebook design, K-means algorithm, Initial codebook generation.

1. **Introduction.** Vector quantization (VQ) [1] is one of widely used techniques for multimedia compression [2] and clustering analysis [3]. A K-Level n-dimensional vector quantiser Q can be defined as a mapping from the n-dimensional Euclidean space R^n into a finite set C, that is, $Q : R^n \to C = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_K\}$, where the set C is called the codebook, K is the codebook size, and $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \ldots, y_{in})^{\mathtt{T}}$, $1 \le i \le K$, are called the codewords. Based on this quantizer, any n-dimensional input vector $\boldsymbol{v} = (v_1, v_2, ..., v_n)^{\mathtt{T}}$ can be quantized into a codeword in C, that is,$Q(\boldsymbol{v}) \in \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_K\}$. The quantization should be based on a suitable distance measure or distortion metric. That is to say, among all codewords in C, the quantization result $Q(\boldsymbol{v})$ of $\boldsymbol{v}$ should be the codeword $\boldsymbol{y}_j$ with the smallest distortion to $\boldsymbol{v}$. The most commonly used distortion measure is the squared error, assume that Q is designed from the training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$,

where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^{\mathsf{T}}$, $1 \leq i \leq M$, then we can quantify the performance of Q by the following average distortion

$$D = E[d(\boldsymbol{.}, Q(\boldsymbol{.})] = \frac{1}{M} \sum_{i=1}^{M} \|\boldsymbol{x}_i - Q(\boldsymbol{x}_i)\|^2 \tag{1}$$

VQ is one of popular image compression techniques because of its simpler decoding structure and it can achieve high compression ratio while maintaining acceptable image quality. Obviously, the compression performance is determined by the quality of the predesigned codebook. Thus an optimal codebook is required to be generated from a training set in advance. The task of optimal vector quantizer design is to find the codebook that minimizes the average distortion over all possible codebooks. In general, an optimal vector quantizer should satisfy two conditions: (1) the nearest neighbor condition, i.e., each training vector should be assigned to the codeword that is closest to it; (2) the best codebook condition, i.e., each codeword must be the centroid of the training vectors that are assigned to it. The above two optimality conditions form an scheme named K-means algorithm for the design of a locally optimal codebook with iterative codebook updating. It is also called the generalized Lloyd algorithm (GLA) or the Linde-Buzo-Gray (LBG) algorithm [4].

It is known that GLA can only converge to a locally optimal codebook, and both the convergence speed and the quality of the generated codebook depend on the initial codebook. Thus, many researchers [5] have been focusing their attention on enhancing the performance of GLA so as to improve either the quality or the speed of VQ. To improve the quality of VQ, most scholars have tried to use population-based meta-heuristics to find a better solution for the codebook generation problem [6, 7, 8] than standard GLA and its variants. However, the cost of population-based meta-heuristics is too high to solve the codebook generation problem online. As the image size and the demand for online processing keep increasing, the speed has become a critical issue in codebook generation. To improve the speed of VQ, many researchers have paid particular attention to the problem of speeding up the codebook generation. These approaches can be divided into three categories. The first is to use more efficient codebook structures to reduce the time required to assign training vectors to codewords such as tree-structured vector quantization (TSVQ) [9]. The second is to reduce the number of comparisons required to assign training vectors to codewords to which they belong [10, 11]. The third is to use a new codeword updating step other than the conventional centroid-based updating step [12, 13].

Since both the convergence speed and the performance of the converged codebook depend on the initial codebook, this paper aims to generate a better initial codebook. A good initial codebook generation method can be combined with any above-mentioned technique to further improve the performance of VQ. In the past, many algorithms have been proposed to obtain a good initial codebook, including the well known splitting, pruning, pairwise nearest neighbor design (PNN), random initialization and the maximum distance initialization [14]. However, all these initialization techniques do not consider the features of each training vector. Therefore, in this paper, we propose a simple and efficient initialization technique for the K-means algorithm by classifying the input vectors into eight subsets with a simple edge classifier and then proportionally and randomly selecting several initial codewords from each subset to obtain the initial codebook. Experimental results demonstrate that, compared with the conventional and modified K-means algorithms with random selection initialization, our initialization technique converges to a better locally optimal codebook with a fast convergence speed.

The rest of this paper is organized as follows. In Section 2, we briefly introduce conventional and modified K-means algorithms. Section 3 states the proposed scheme in detail. The simulation results and analysis are given in Section 4. Finally, we conclude the whole paper in Section 5.

2. **Conventional and Modified K-Means Algorithms.** Assume the training set X is composed of M n-dimensional vectors, i.e., $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$, then the conventional K-means algorithm [4] for the design of a codebook $C = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_K\}$ of size K is as follows.

Step 1. Initialization: Iteration number $m = 0$; codebook at iteration m, $C(m) = \{\boldsymbol{y}_1^{(m)}, \boldsymbol{y}_2^{(m)}, \ldots, \boldsymbol{y}_K^{(m)}\}$; convergence threshold e.

Step 2. Partitioning: Find the nearest-neighbor partition $V_j^{(m)} = \{\boldsymbol{x}_i \in X : Q^{(m)}(x_i) = \boldsymbol{y}_j^{(m)}\}$, $j = 1, 2, \ldots, K$. Here, $Q^{(m)}$ denotes the vector quantizer defined as: $Q^{(m)}(\boldsymbol{v}) = \boldsymbol{y}_j^{(m)}$ if $d(\boldsymbol{v}, \boldsymbol{y}_j^{(m)}) \leq d(\boldsymbol{v}, \boldsymbol{y}_l^{(m)})$, $l = 1, 2, \ldots, K$.

Step 3. Codebook Updating: Update codewords $C^{(m)} = \{\boldsymbol{y}_j^{(m)}, j = 1, 2, \ldots, K\}$ to $C^{(m+1)} = \{\boldsymbol{y}_j^{(m+1)}, j = 1, 2, \ldots, K\}$ as

$$\boldsymbol{y}_j^{(m+1)} = \frac{1}{|V_j^{(m)}|} \sum_{\boldsymbol{v} \in V_j^{(m)}} \boldsymbol{v} \tag{2}$$

that is, $\boldsymbol{y}_j^{(m+1)}$ is the centroid of the partition $V_j^{(m)}$, where $|V_j^{(m)}|$ stands for the operation to get the number of elements in $V_j^{(m)}$.

Step 4. Termination Check: Stop if $|d_{m+1} - d_m|/d_{m+1} \leq e$, where d_{m+1} is the average distortion after $m + 1$ iterations defined as

$$d_{m+1} = \frac{1}{M} \sum_{i=1}^{M} \|\boldsymbol{x}_i - Q^{(m+1)}(\boldsymbol{x}_i)\|^2 \tag{3}$$

Otherwise, replace m by $m + 1$ and go to Step 2.

The modified K-means algorithm proposed by Lee et al. [12] is almost the same as the conventional K-means algorithm except for a modification at the codebook updating step. They update the current codeword $\boldsymbol{y}_j^{(m)}$ at iteration m to the new codeword $\boldsymbol{y}_j^{(m+1)}$ at iteration $m + 1$ as

$$\boldsymbol{y}_j^{(m+1)} = \boldsymbol{y}_j^{(m)} + s \times ((\frac{1}{|V_j^{(m)}|} \sum_{\boldsymbol{v} \in V_j^{(m)}} \boldsymbol{v}) - \boldsymbol{y}_j^{(m)}) \tag{4}$$

where s is a scale factor. Based on the squared-error distance measure, Lee et al. [12] have shown experimentally that the modified K-means algorithm converges slower in comparison to the conventional K-means algorithm when $s < 1$. When $1 < s < 2$, it converges faster and results in better performance. When $s > 2$, the algorithm either does not converge, or converges very slowly with poor performance. When $s = 1$, the modified K-means algorithm is just the same as the conventional K-means algorithm. The best results are found when the scale factor is set to a fixed value of $s = 1.8$.

Although the use of a scaled-update as in Eq. (4) can accelerate the convergence, the use of a fixed scaling for the entire range of iterations results in the use of step sizes larger than the corresponding centroid-update at iterations closer to convergence and causes undesirably high perturbations of the codewords which are otherwise converging to some optimal configuration. This in turn has the effect of increasing the number of iterations required to converge as well as perturbing the codebook convergence to a poorer local

optimum. Thus, Paliwal and Ramasubramanian [13] proposed the use of a variable scale factor s in the codeword update which varies as a function of the iteration m and is inversely proportional to m as follows:

$$s = 1.0 + \frac{x}{x+m} \tag{5}$$

where $x > 0$. In this equation, $s = 2$ when $m = 0$, and $s = 1$ when $m = \infty$. thus, it satisfies the aforementioned conditions. To see the effect of variable x used in the scale factor equation, Paliwal and Ramasubramanian have studied the algorithm with various values of x. According to their results, they finally adopt $x = 9$.

3. **The Proposed Scheme.** Vector quantization (VQ) allows the modeling of probability density functions by the distribution of prototype vectors. It works by dividing a large set of vectors into groups having approximately the same number of points closest to them. Each group is represented by its centroid point. The density matching property of vector quantization is powerful, especially for identifying the density of large and high-dimensioned data. The feature distribution of the training set is very important for initial codebook generation. Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction. Our algorithm therefore considers classifying the training vectors into eight classes according to their edge orientations. The basic idea is to build a histogram with the directions of the gradients of the edges. It is possible to detect edges in an image but here we are interested in the detection of the angles. Assume the training set is $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$, the proposed scheme is illustrated in Fig.1. The detailed scheme can be expressed as follows:

Step 1: First, each training vector $\boldsymbol{x}_i (i = 1, 2, ..., M)$ is input into the edge classifier, and an index $t_i \in \{1, 2, ..., 8\}$ is output to denote which class the training vector belongs to.

Step 2: Then, we collect all the training vectors belonging to the same class to generate a subset, and thus we have 8 subsets P_j of sizes $s_j (j = 1, 2, \ldots, 8)$ respectively, where $s_1 + s_2 + \ldots + s_8 = M$.

Step 3: Afterwards, we initialize Ks_j/M initial codewords from each subset P_j based on random selection, thus we can in total obtain K initial codewords.

Step 4: Finally, the modified K-means algorithm in [13] is performed to generate the final codebook.

During the iteration steps, if an empty cell occurs, we just judge the classes that all current codewords belong to and find the class with the least number of codewords, and randomly select a training vector from this class as a new centroid.

Now, we describe how to classify the training vectors based on our edge classifier. Inspired by the Structured Local Binary Kirsch Pattern (SLBKP) in [14] that adopts eight 3×3 Kirsch templates to denote eight edge directions, we propose following eight 4×4 templates for edge classification as shown in Fig.2.

Assume the input image X is segmented into non-overlapping 4×4 blocks, the edge classification can be described as follows: First, we perform eight 4×4 edge orientation templates on each 4×4 block $x(p, q), 0 \leq p < 4, 0 \leq q < 4$, obtaining an edge orientation vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_8)$ with its components $v_i (1 \leq i \leq 8)$ being calculated as follows:

$$v_i = |\sum_{p=0}^{3} \sum_{q=0}^{3} [x(p, q) \cdot e_i(p, q)]| 1 \leq i \leq 8 \tag{6}$$

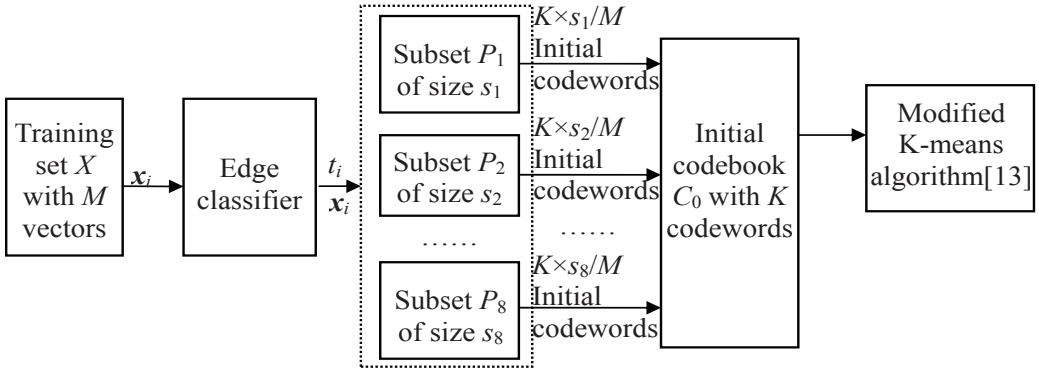

FIGURE 1. The block diagram of the proposed edge classified K-means algorithm.

$$\mathbf{E}_1=\begin{bmatrix}-4&2&2&2\\-4&0&0&2\\-4&0&0&2\\-4&2&2&2\end{bmatrix},\mathbf{E}_2=\begin{bmatrix}2&2&2&2\\0&0&2&2\\-4&-4&0&2\\-4&-4&0&2\end{bmatrix},\mathbf{E}_3=\begin{bmatrix}2&2&2&2\\2&0&0&2\\2&0&0&2\\-4&-4&-4&-4\end{bmatrix}$$

$$\mathbf{E}_4=\begin{bmatrix}2&2&2&2\\2&2&0&0\\2&0&-4&-4\\2&0&-4&-4\end{bmatrix},\mathbf{E}_5=\begin{bmatrix}2&2&2&-4\\2&0&0&-4\\2&0&0&-4\\2&2&2&-4\end{bmatrix},\mathbf{E}_6=\begin{bmatrix}2&0&-4&-4\\2&0&-4&-4\\2&2&0&0\\2&2&2&2\end{bmatrix}$$

$$\mathbf{E}_7=\begin{bmatrix}-4&-4&-4&-4\\2&0&0&2\\2&0&0&2\\2&2&2&2\end{bmatrix},\mathbf{E}_8=\begin{bmatrix}-4&-4&0&2\\-4&-4&0&2\\0&0&2&2\\2&2&2&2\end{bmatrix}$$

FIGURE 2. Eight 4×4 templates for edge classification.

Where $e_i(p,q)$ denotes the element at the position (p,q) of $\mathtt{E}_i$. Thus, an input block $x(p,q)$ is classified into the j-th category if

$$j=\arg\max_{1\le i\le 8} v_i \tag{7}$$

Thus, we can classify each training vector into one of eight categories according to its edge orientation.

4. **Simulations.** To demonstrate the performance of the proposed edge classified K-means algorithm, we compared our scheme with the traditional K-means algorithm (KMeans), the modified K-means algorithm with the fixed scale value $s=1.8$ [6] (MKMeans_F) and the modified K-means algorithm with a variable scale value and $x=9$ [7] (MKMeans_V). In our experiments, we used two 512×512 monochrome images with 256 gray levels, Lena and Peppers. We segmented each image into 16384 blocks, and each block is of size 4×4. We tested the performance for different codebook sizes of 256, 512, and 1024. The quality of the compressed images is evaluated by PSNR (Peak Signal to Noise Ratio). Let the size of the original 256-grayscale image be 512×512, x_{ij} be the original pixel value at the

TABLE 1. Performance comparison with random selection initialization. (CBsize: codebook Size; itr: number of iterations).

CBsize		**256**		**512**		**1024**	
Performance		**PSNR(dB)**	**itr**	**PSNR(dB)**	**itr**	**PSNR(dB)**	**itr**
Lena	**KMeans: Best**	30.447	25	31.293	29	32.138	27
	KMeans: Ave	30.379	36	31.237	38	32.083	34
	MKMeans_F: Best	30.495	22	31.453	23	32.439	20
	MKMeans_F: Ave	30.445	26	31.420	27	32.387	26
	MKMeans_V: Best	30.470	16	31.420	14	32.452	17
	MKMeans_V: Ave	30.436	23	31.393	24	32.383	22
	Our: Best	30.501	18	31.463	20	32.441	15
	Our: Ave	**30.450**	**22**	**31.420**	**23**	**32.391**	**20**
Peppers	**KMeans: Best**	29.863	32	30.591	24	31.368	19
	KMeans: Ave	29.799	43	30.540	37	31.313	29
	MKMeans_F: Best	29.956	25	30.789	25	31.712	18
	MKMeans_F: Ave	29.917	37	30.714	33	31.625	24
	MKMeans_V: Best	29.912	20	30.758	19	31.732	16
	MKMeans_V: Ave	29.861	30	30.710	24	31.593	20
	Our: Best	29.962	16	30.793	17	31.735	16
	Our: Ave	**29.921**	**27**	**30.718**	**23**	**31.630**	**19**
Baboon	**KMeans: Best**	23.23	35	23.893	31	24.616	20
	KMeans: Ave	23.21	44	23.874	39	24.589	26
	MKMeans_F: Best	23.267	30	23.954	26	24.764	19
	MKMeans_F: Ave	23.245	36	23.938	30	24.745	23
	MKMeans_V: Best	23.253	23	23.946	20	24.746	15
	MKMeans_V: Ave	23.231	29	23.927	25	24.723	18
	Our: Best	23.255	23	23.965	20	24.763	14
	Our: Ave	**23.246**	**29**	**23.940**	**24**	**24.747**	**17**

position (i, j) and x'_{ij} be the decoded pixel value at the position (i, j), then PSNR can be defined as follows:

$$\texttt{PSNR} = 10 \log_{10} \frac{255^2}{\frac{\sum_{i=0}^{511} \sum_{j=0}^{511} (x_{ij} - x'_{ij})^2}{512 \times 512}} \tag{8}$$

Because the random selection is adopted for each algorithm, the performance is averaged over ten runs.

All the algorithms are terminated when the ratio of the mean squared error difference between two iterations to the mean squared error of the current iteration is within 0.0001 or 0.01%. In Table 1, the PSNR values and the numbers of iterations for different images with different codebook sizes are shown, where "Best" and "Ave" denote the best and the average results over ten runs respectively. From Table 1, we can see that, if the random selection technique is used, our scheme requires the least average number of iterations than other algorithms and can also get better codebooks than other algorithms on average.

5. **Conclusions.** This paper presents an improved K-means algorithm for image vector quantization. The main idea is to classify the training vectors into 8 categories based on the edge orientation of each vector, and then randomly select initial codewords from each category with the number of codewords proportional to the number of vectors in each category. The experimental results based on three test images show that our algorithm can converge to a better locally optimal codebook with a faster convergence speed. Future

work will concentrate on how to select more other features and classify them to obtain more representative features for color image retrieval.

Acknowledgments. This work was partly supported by the Public Good Research Project of Science and Technology Program of Zhejiang Province, China. (NO.2016C31097 and NO.2015C33230), the Science and Technology Project of QuZhou, Zhejiang, China. (NO.2015Y005).

REFERENCES

[1] A. Gersho, and R. M. Gray, Vector quantization and signal compression, Springer Science & Business Media, 2012.

[2] F. D. V. R. Oliveira, H. L. Haas, J. G. R. C. Gomes, and A. Petraglia, CMOS imager with focal-plane analog image compression combining DPCM and VQ, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no.5, pp. 1331–1344, 2013.

[3] F. X. Yu, H. Luo, and Z. M. Lu, Colour image retrieval using pattern co-occurrence matrices based on BTC and VQ, *Electronics Letters*, vol. 47, no.2, pp. 100–101, 2011.

[4] Y. Linde, A. Buzo, and R. M. Gray, An algorithm for vector quantizer design, *IEEE Transactions on Communications*, vol. 28, no.1, pp. 84–95, 1980.

[5] A. Vasuki, and P. Vanathi, A review of vector quantization technique, *IEEE Potentials*, vol. 25, no.4, pp. 39-C47, 2006.

[6] M. H. Horng, and T. W. Jiang, The artificial bee colony algorithm for vector quantization in image compression, *The 4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, pp. 319–323, 2011.

[7] H. A. S. Leitao, W. T. A. Lopes, and F. Madeiro, PSO algorithm applied to codebook design for channel-optimized vector Quantization, *IEEE Latin America Transactions*, vol. 13, no. 4, pp. 961-C967, 2015.

[8] S. Alkhalaf, O. Alfarraj, and A. M. Hemeida, Fuzzy-VQ image compression based hybrid PSOGSA optimization algorithm, *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–6, 2015.

[9] C. C. Chang, Y. C. Li, and J. B. Yeh, Fast codebook search algorithms based on tree-structured vector quantization, *Pattern Recognition Letters*, vol. 27, no. 10, pp. 1077C-1086, 2006.

[10] Z. Pan, K. Kotani, and T. Ohmi, An improved full-search-equivalent vector quantization method using the law of cosines, *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 247-C250, 2004.

[11] J. Z. C. Lai, Y. C. Liaw, and J. Liu, A fast VQ codebook generation algorithm using codeword displacement, *Pattern Recognition*, vol. 41, no. 1, pp. 315-C319, 2008.

[12] D. Lee, S. Baek, and K. Sung, Modified K-means algorithm for vector quantizer design, *IEEE Signal Processing Letters*, vol. 4, no.1, pp. 2–4, 1997.

[13] K. K. Paliwal, and V. Ramasubramanian, Comments on modified K-means algorithm for vector quantizer design, *IEEE Transactions on Image Processing*, vol. 9, no.11, pp. 1964–1967, 2000.

[14] I. Katsavounidis, C. C. J. Kuo, and Z. Zhang, A new initialization technique for generalized Lloyd iteration, *IEEE Signal Processing Letters*, vol. 1, no.10, pp. 144–146, 1994.

[15] G. Y. Kang, S. Z. Guo, D. C. Wang, L. H. Ma, and Z. M. Lu, Image retrieval based on structured local binary kirsch pattern, *IEICE Transactions on Information and Systems*, vol. 96, no.5, pp. 1230–1232, 2013.