Journal of Information Hiding and Multimedia Signal Processing ©2018 ISSN 2073-4212
Ubiquitous International Volume **9**, Number **5**, September **2018**

Improved K-means Algorithm Using Initialization Technique Based on Edge-Mean Grid for Image Vector Quantizer Design

Zhi-Min Fang and Xiao-Dan Jiang

College of Electrical and Information Engineering
Quzhou University
Quzhou, 324000, P. R. China
fzim@21cn.com

Zhe-Ming Lu*

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

Received February 2018; Revised March 2018

Abstract. *The K-means algorithm is famous as a traditional data clustering technique, which is also effective in the field of codebook design for Vector Quantization (VQ) based data compression. The main disadvantage of the conventional K-means algorithm is that the convergence speed and the quality of the generated codebook depend on the initial codebook. Many researchers have been devoted to improving the initial codebook, but their algorithms either do not make full use of the features of the training vectors or need high extra computational load. This paper presents an efficient initialization technique based on the edge-mean grid generated by an edge classifier as well as even grouping of sorted means. On the one hand, the training vectors are sorted in the descending order according to their mean values. On the other hand, the training vectors are classified into 8 categories based on the edge classifier, and the number of initial codewords generated from each category is proportional to the number of training vectors in this category. The mean-sorted training vectors in each category is evenly grouped into several groups with the number of groups equaling the number of codewords to be generated from the category. Thus, an edge-mean grid is obtained with many groups, and each group generates one initial codeword that is just the middle training vector in the group. Experimental results show that, compared with the random selection and the existing norm-sorted and mean-sorted grouping strategies, our initialization technique converges to a better codebook with a faster convergence speed.*
Keywords: Vector quantization, Codebook design, K-means algorithm, Initial codebook generation, Image compression, Edge-mean grid

1. **Introduction.** Vector quantization (VQ) [1] is a popular lossy image compression technique [2] because it has a simple decoding structure and can achieve high compression ratio while maintaining acceptable reconstructed quality. Recently, VQ has also been successfully used in other fields such as feature extraction [3] and data hiding [4]. In the n-dimensional Euclidean space R^n, a vector quantizer Q can be viewed as a mapping from R^n into one of its subset named codebook $C = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_K\}$, where $\boldsymbol{y}_i = (y_{i1}, y_{i2}, ..., y_{in})^T$.

The squared error $d(\boldsymbol{u}, \boldsymbol{v}) = ||\boldsymbol{u} - \boldsymbol{v}||^2$ is often used to measure the distance between two n-dimensional vectors. In practice, the vector quantizer Q is often designed from a training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_M\}$, and the performance can be evaluated by the average distortion $D = E[d(\boldsymbol{v}, Q(\boldsymbol{v}))], \boldsymbol{v} \in X$.

The performance of VQ entirely depends on the codebook size and the codebook design algorithm. Thus, it is an essential key step for VQ to find the codebook that minimizes the average distortion over all possible codebooks. For an optimal vector quantizer, each training vector should be mapped to the codeword that is closest to it, and each codeword should be the centroid of the training vectors that are mapped to it. The traditional K-means algorithm also called the generalized Lloyd algorithm (GLA) [5] is just based on these two requirements iteratively to generate a locally optimal codebook.

Although the original K-means algorithm is simple and easy to understand, it can only converge to a locally optimal codebook. Thus, many researchers [6] have been devoted to improving the quality based on population-based meta-heuristics schemes [7]. Furthermore, not only the convergence speed but also the quality of the generated codebook greatly depend on the initial codebook. To speed up the speed of VQ, some scholars have proposed using more efficient codebook structures or codeword reduction methods [8, 9, 10, 11, 12]. To improve the quality, many researchers have been devoted to generating a better initial codebook. The conventional algorithms for selecting an initial codebook are the random selection algorithm, the splitting algorithm [5] and the pairwise nearest neighbour (PNN) algorithm [13] and the maximum distance initialization scheme [14]. In the past decade, some new initialization techniques have been proposed. In 2008, Chen et al. presented a norm-sorted grouping strategy [15], where the training vectors are sorted according to the norm of each vector component. In 2010, Chen and Li [16] proposed a high-quality initial codebook method of VQ based on mean grouping. In 2014, Mirzaei et al. [17] proposed an initial codebook design method using subtracting clustering. However, most existing initialization schemes either do not make full use of the features of training vectors or need high extra computation load. In this Letter, for image vector quantizer design, we present a simple and efficient initialization scheme for the K-means algorithm based on the edge-mean grid. The main idea it to classify the training vectors into 8 categories by a edge classifier and sort the training vectors in each category according to their mean values. Experimental results show that, our approach can generate a better codebook than random initialization and the existing norm-sorted or mean-sorted grouping strategy with a faster speed.

2. **Related works.**

2.1. **Traditional K-means Algorithms.**

In VQ, the K-means algorithm is the first efficient codebook design scheme. The VQ codebook is generated from the training vectors after the iterative processing. The detailed steps of the original K-means algorithm [5] can be described as follows.

Input: The training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_M\}$ of size M.

Output: The codebook $C = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_K\}$ of size K.

Step 1. Generate an initial codebook $C^{(0)} = \{\boldsymbol{y}_1^{(0)}, \boldsymbol{y}_2^{(0)}, ..., \boldsymbol{y}_K^{(0)}\}$ of size K. Set the iteration counter $m = 0$ and the initial average distortion $D_{-1} = \infty$. Set the convergence threshold ε.

Step 2. For each training vector $\boldsymbol{v}$, find its closest codeword $\boldsymbol{y}_j^{(m)}$ in the current codebook $C^{(m)}$. Thus we can obtain K partitions $V_j^{(m)} = \{\boldsymbol{v} \in X | Q^{(m)}(\boldsymbol{v}) = \boldsymbol{y}_j^{(m)}\}, j = 1, 2, \ldots, K$.

Here, $Q^{(m)}$ denotes the vector quantizer defined as: $Q^{(m)}(\boldsymbol{v}) = \boldsymbol{y}_j^{(m)}$ if $d(\boldsymbol{v}, \boldsymbol{y}_j^{(m)}) \leq d(\boldsymbol{v}, \boldsymbol{y}_i^{(m)}), i = 1, 2, \ldots, K$.

Step 3: Calculate the current average distortion D_m as follows.

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

Step 4. Calculate the centroid of each partition to obtain a new codebook $C^{(m+1)} = \{\boldsymbol{y}_j^{(m+1)}, j = 1, 2, \ldots, K\}$ as

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

where $|V_j^{(m)}|$ stands for the number of elements in $V_j^{(m)}$.

Step 5. If the relative improvement $|D_m - D_{m-1}|/D_m \leq \varepsilon$, terminate the algorithm with $C = C^{(m+1)}$. Otherwise, replace m by $m + 1$ and go to Step 2.

To speed up the K-means algorithm, Lee et al. [15] proposed a modification at the codebook updating step. They update the current codeword as follows

$$\boldsymbol{y}_j^{(m+1)} = \boldsymbol{y}_j^{(m)} + s \times ((\frac{1}{|V_j^{(m)}|} \sum_{\boldsymbol{v} \in V_j^{(m)}} \boldsymbol{v}) - \boldsymbol{y}_j^{(m)}) \tag{3}$$

where s is a scale factor. Based on the squared-error distance measure, the best scale factor is experimentally found to be $s = 1.8$. To further improve the algorithm, Paliwal and Ramasubramanian [19] proposed using a variable scale factor other than a fixed one. This variable scale factor s is inversely proportional to m as follows:

$$s = 1.0 + \frac{x}{x + m} \tag{4}$$

where $x > 0$. According to the experimental results, they finally utilize $x = 9$.

2.2. **Typical Existing Initialization Techniques.** From the detailed steps of the K-means algorithm, it can be observed that the performance of the obtained codebook depends on the initial codebook. Thus, it is important to find a good initial codebook. In this subsection, we introduce some typical existing initialization techniques.

The random selection algorithm is simply to select K codewords from the training vectors of size M using a random selection strategy, where $M \gg K$. This algorithm is very simple and has a low complexity. However, the performance of the initial codebook is rather poor.

The main idea of the splitting algorithm [5] is to split the current small codebook into a larger codebook. At first, there is only one codeword in the codebook, which is the centroid of the whole training set. In each splitting stage, each codeword in the codebook $\{\boldsymbol{y}_i, i = 1, 2, \ldots, N\}$ is split into two close codewords $\boldsymbol{y}_i + \varepsilon$ and $\boldsymbol{y}_i - \varepsilon$, where ε is a fixed perturbation vector. Then the set $\{\boldsymbol{y}_i + \varepsilon, \boldsymbol{y}_i - \varepsilon; i = 1, 2, \ldots, N\}$ becomes the new codebook of size $2N$ in the next splitting phase. This process is complete until the codebook size K is reached.

The PNN algorithm [13] performs an inverse process compared with the splitting scheme. First, each cluster has only one training vector. Then the PNN algorithm merges two closest clusters together at a time and the codeword of the new cluster is the centroid of the new cluster. The algorithm is complete until the desired codebook

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

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

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

FIGURE 1. Eight 4×4 templates for edge classification.

size K is reached. Both the PNN algorithm and the splitting algorithm have a very high computational overhead.

The main idea of maximum distance initialization [14] is to find the training vectors that are most far apart from each other. This scheme first calculates the norms of all vectors in the training set, and chooses the vector with the maximum norm as the first codeword. Then, calculate the distance of all training vectors from the first codeword, and select the vector with the largest distance as the second codeword. With a codebook of size i, $i = 2, 3, \ldots$, compute the distance between any remaining training vector $\boldsymbol{v}$ and all existing codewords and call the smallest one the distance between $\boldsymbol{v}$ and the codebook. Then, the training vector with the largest distance from the codebook is chosen to be the $(i + 1)$-th codeword. The procedure stops until the codebook size K is reached. This scheme is of high complexity for a large codebook.

In 2008, Chen et al. presented a norm-sorted grouping strategy [15], where the training vectors are sorted according to their norm values. The ordered vectors are then partitioned into K groups evenly. The initial codebook is composed of the centroid of each group. This method has worse performance than the random method in the case of large codebook size.

In 2010, Chen and Li [16] proposed an initial VQ codebook design method, where the training vectors are transformed into the Hadamard domain. Then the transform vectors are sorted according to their first element. In fact, the first element is equivalent to the mean value. The ordered transform vectors are then partitioned into groups with equal sizes. The middle vector of each ordered group are selected to construct the initial codebook.

3. **Proposed K-means Algorithm Based on Edge-Mean Grid.**

3.1. **Edge Orientation Classification.** Before describing our algorithm, we first introduce our edge classifier. Inspired by the Structured Local Binary Kirsch Pattern (SLBKP) proposed in [20] that adopts eight 3×3 Kirsch templates to discriminate eight edge orientations, we propose following eight 4×4 templates for edge orientation classification as shown in Fig.1:

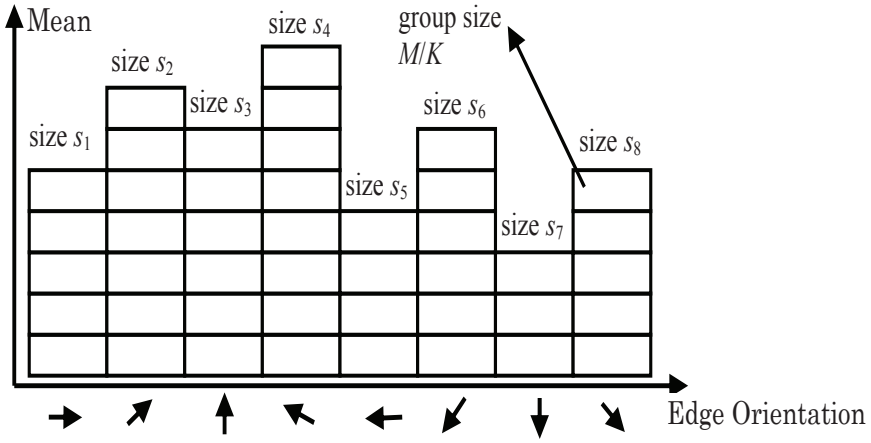

FIGURE 2. The proposed edge-mean grid.

Given a block of 4×4, e.g., $x(p,q), 0 \leq p < 4, 0 \leq q < 4$, an edge orientation vector $\boldsymbol{v} = (v_1, v_2, ..., v_8)$ can be calculated as follows:

$$v_i = |\sum_{p=0}^{3}\sum_{q=0}^{3}[x(p,q) \cdot e_i(p,q)]| 1 \leq i \leq 8 \tag{5}$$

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

$$j = \arg\max_{1 \leq i \leq 8} v_i \tag{6}$$

Thus, based on the above classifier, we can classify each training vector into one of eight categories according to its edge orientation.

3.2. **Edge-Mean Grid Generation.** In fact, the feature distribution of the training set has a great effect on the initial codebook generation. To make full use of the features, we consider generating a edge-mean grid for initial codebook generation. This grid takes into account two kinds of features, i.e., edge orientation and brightness (denoted by mean). We classify the training vectors into eight categories according to their edge orientations and then sort the training vectors in each category based on their mean values. Finally, the sorted training vectors in each category are grouped evenly (each group is of size M/K) to generate the edge-mean grid as shown in Fig. 2.

3.3. **Algorithm Description.** Now we turn to describing our algorithm. Assume the training set is $X = \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_M$, the detailed steps of our scheme are as follows. First, each training vector $\boldsymbol{x}_i$ is input into the edge classifier, and an index $t_i \in 1, 2, \ldots, 8$ is output to denote which class the training vector belongs to. 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 $s_1 + s_2 + \ldots + s_8 = M$. Afterwards, we initialize $K \times s_j/M$ initial codewords from each subset P_j based on mean-sorted grouping strategy as follows: (1) calculate the mean value of each training vector in subset P_j; (2) sort the training vectors in P_j in the descending order of their means; (3) divide the sorted subset P_j into $K \times s_j/M$ groups evenly as shown in Fig.2; (4)adopt the middle vector of each group as the initial codeword. Finally, we collect all middle vectors from different groups, we can in total obtain K initial codewords. Then the modified

K-means algorithm in [19] is performed on the generated K initial codewords to generate the final codebook.

4. **Experimental Results and Discussion.** To test and evaluate the performance of the proposed edge-mean grid based K-means algorithm, we compared it with the traditional K-means algorithm (KMeans), the norm-ordered grouping based algorithm (NOKMeans) [15], the mean-ordered grouping based algorithm (MOKMeans) [16]. For KMeans, we use the random selection technique, the performance is averaged over ten runs. In our experiments, we used three 512×512 monochrome images with 256 gray levels, Lena, Peppers and Baboon. 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 mean squared error (MSE).

All the algorithms are terminated when the ratio of the MSE difference between two iterations to the MSE of the current iteration is within 0.0001 or 0.01%. In Tables 1-3, the PSNR values and the numbers of iterations with different codebook sizes are shown for Lena, Peppers and Baboon images respectively. From Tables 1-3, we can see that, our scheme requires the least average number of iterations than other algorithms and can also get better codebooks than other algorithms.

Table 1. Performance comparison for Lena image. (itr: number of iterations)

Codebook size	256		512		1024	
Performance	MSE	itr	MSE	itr	MSE	itr
KMeans	60.415	39	49.213	40	40.624	34
NOKMeans[15]	59.344	42	49.104	45	41.790	41
MOKMeans[16]	59.426	32	48.833	39	40.507	31
Our	**58.594**	**27**	**47.249**	**27**	**38.097**	**19**

Table 2. Performance comparison for Peppers image. (itr: number of iterations)

Codebook size	256		512		1024	
Performance	MSE	itr	MSE	itr	MSE	itr
KMeans	68.358	45	58.135	39	48.795	29
NOKMeans[15]	68.057	45	58.234	49	49.564	35
MOKMeans[16]	68.330	44	57.468	32	48.718	28
Our	**66.365**	**42**	**55.649**	**27**	**44.448**	**20**

Table 3. Performance comparison for Baboon image. (itr: number of iterations)

Codebook size	256		512		1024	
Performance	MSE	itr	MSE	itr	MSE	itr
KMeans	314.68	46	266.56	39	227.28	30
NOKMeans[15]	310.11	47	267.62	61	229.71	29
MOKMeans[16]	310.20	43	265.32	41	225.54	28
Our	**308.13**	**28**	**263.19**	**22**	**219.06**	**19**

5. **Conclusion.** This paper presents an edge-mean grid based K-means algorithm for image vector quantization. The main idea is to classify the training vectors into 8 categories based on the edge orientation, and then select initial codewords from each category based on mean sorting and grouping with the number of codewords proportional to the number of training 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.

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, R.M. Gray, Vector quantization and signal compression, Springer Science and Business Media, 2012.
[2] F. D. V. R. Oliveira, H. L. Haas, J. G. R. C. Gomes, 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, 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] C. Qin, Y. C. Hu, Reversible data hiding in VQ index table with lossless coding and adaptive switching mechanism, *Signal Processing*, vol.129, no.1, pp. 48–55, 2016.
[5] Y. Linde, A. Buzo, R. M. Gray, An algorithm for vector quantizer design, *IEEE Transactions on Communications*, vol. 28, no.1, pp. 84–95, 1980.
[6] A. Vasuki, P. Vanathi, A review of vector quantization technique, *IEEE Potentials*, vol. 25, no.4, pp. 39–47, 2006.
[7] H. A. S. Leitao, W. T. A. Lopes, F. Madeiro, PSO algorithm applied to codebook design for channel-optimized vector Quantization, *IEEE Latin America Transactions*, vol. 13, no. 4, pp. 961–967, 2015.
[8] 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. 1077–1086, 2006.
[9] 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. 315-319, 2008.
[10] J. S. Pan, F. R. McInnes, M. A. Jack, Fast clustering algorithms for vector quantization, *Pattern Recognition*, vol. 29, no. 3, pp.511–518, 1996.
[11] J. S. Pan, K. C. Huang, A new vector quantization image coding algorithm based on the extension of the bound for Minkowski metric, *Pattern Recognition*, vol. 31, no. 11, pp. 1757–1760, 1998
[12] T. C. Lu, C. Y. Chang, A survey of VQ codebook generation, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 1, no. 3, pp. 190–203, 2010.
[13] W. H. Equitz, A new vector quantization clustering algorithm, *IEEE Transactions on Acoustic Speech Signal Processing*, vol. 37, no.10, pp. 1568–1575, 1989.
[14] I. Katsavounidis, C. C. J. Kuo, Z. Zhang, A new initialization technique for generalized Lloyd iteration, *IEEE Signal Processing Letters*, vol. 1, no.10, pp. 144–146, 1994.
[15] S. X. Chen, F. W. Li, W. L. Zhu, T.Q. Zhang, Initial codebook algorithm of vector quantization, *IEICE Transactions on Information and Systems*, E91-D, no.8, pp. 2189–2191, 2008.
[16] S. X. Chen, F. W. Li, Initial codebook method of vector quantisation in Hadamard domain, *Electronics Letters*, vol. 46, no.9, pp. 630–631, 2010.
[17] B. Mirzaei, N. P. Hossein, A. M. Dariush, An effective codebook initialization technique for LBG algorithm using subtractive clustering, *Iranian Conference on Intelligent Systems (ICIS)*, pp. 1–5, 2014.
[18] D. Lee, S. Baek, K. Sung, Modified K-means algorithm for vector quantizer design, *IEEE Signal Processing Letters*, vol. 4, no.1, pp. 2–4, 1997.
[19] K. K. Paliwal, 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.

[20] G. Y. Kang, S. Z. Guo, D. C. Wang, L. H. Ma, 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.