IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 14, Issue 3 (Sep. - Oct. 2013), PP 89-98
www.iosrjournals.org

Performance Comparison of K-means Codebook Optimization using different Clustering Techniques

Dr. Tanuja K. Sarode [1], Nabanita Mandal [2]
[1](Associate Professor, Computer Engineering Department, Thadomal Shahani Engineering College, India)
[2](Lecturer, Computer Engineering Department, Thadomal Shahani Engineering College, India)

Abstract : *Vector quantization is a compression technique which is used to compress the image data in the spatial domain. Since it is a lossy technique, so maintaining the image quality and the compression ratio is a difficult task. For this, the codebook which stores the image data should be optimally designed. In this paper, the K-means algorithm is used to optimize the codebook. We have generated the codebooks from LBG, KPE, KFCG and Random Selection Methods. It is found that the Mean Square Error reduces for every iteration and after a certain number of iterations it stops reducing because the optimal value is reached. We can say that the codebook is optimized at that point. The results show that the codebook obtained from KFCG Algorithm has a least Mean Square Error. This shows that KFCG codebook is more nearer to the optimal point and when applied with K-means algorithm gives the best optimization in comparison with other algorithms.*
Keywords: *Codebook Optimization, Euclidian Distance, K-means Algorithm, Mean Square Error, Vector Quantization*

I. INTRODUCTION

With the growing popularity of the internet, the need for transmission of images has increased. But for transmitting the image quickly, high bandwidth is required. Compression of the image is one way of fast transmission. The aim behind image compression is the reduction of irrelevant and redundant data of the image in order to store or transmit the image in an efficient way. Various lossy and lossless techniques are available for compression [1]. Lossless compression techniques are suitable where each and every minute technical detail of the image is significant. Loss of a single detail would lead to transmission of an improper image. Lossy compression methods are suitable for natural images where minor loss of fidelity is allowed. One such lossy compression technique is Vector Quantization (VQ) [2]. In this technique, the image is represented in the form of vectors. VQ is a mapping function, Q that maps a vector in K-dimensional vector space, R^k into a finite subset of the vector space W containing N distinct vectors [3]. Hence, Q: $R^k \rightarrow W$.

In this technique, the amount of data contained in an image is reduced so that images can be economically transmitted. The image data is stored in a codebook. Each vector present in the codebook is called a codevector. A good codebook is very much essential for VQ. The codebook size is decided and then after applying the different techniques for codebook generation, the Mean Square Error (MSE) is obtained. The distortion obtained from the different codebook generation algorithm varies even if size of the codebook remains same**.** But the minimum error is not obtained. The reason is that the codevectors in the codebook may not have reached their optimal position. When the codebook is optimized, the MSE reaches a value after which it cannot be reduced further.

In literature, vector quantization has been successfully used for image compression. In Pamela C. Cosman et al's paper [4], they have given the fundamental idea of vector quantization and how it can be used for image compression. The process of VQ involves codebook design. A survey of codebook generation techniques has been done by Tzu-Chuen Lu and Ching-Yun Chang. In their paper, LBG, Enhanced LBG, Neural Network based techniques, Genetic Algorithm based techniques etc has been discussed [5]. After the codebook is generated, optimization of the codebook is done using various algorithms. H.B. Kekre and Tanuja K. Sarode has proposed K-means algorithm for optimization of codebook [6]. LBG and KFCG algorithm has been applied by them for generating the codebook and to optimize it they have used K-means algorithm. S. Vimala has done the convergence analysis of various codebook generation techniques when K-means algorithm is applied to them for optimization [7].

In this paper, our main concentration will be on optimization of the codebook. The K-means Algorithm [8] which is a clustering technique, will be used on the codebook for optimization. In this technique, initially, the codevectors are selected randomly. But the initial selection doesn't lead to optimization. So, this process is repeated for many iterations and the MSE value is calculated for each iteration. The MSE reduces in each iteration. After some iterations, the MSE value converges. This point is the optimal point and the optimized codebook is obtained. The K-means Algorithm is applied to the codebook generated by LBG Algorithm [9], KPE Algorithm [10], KFCG Algorithm [11] and Random Selection method. Further, the image is reconstructed back using the optimized codebook.

This paper consists of five sections. In Section II the proposed system has been discussed. Section III contains the description of the algorithms used. Section IV provides the results obtained after application of various techniques. The conclusion is given in Section V.

II. VQ IMAGE COMPRESSION SYSTEM

The VQ image compression system is shown in Fig. 1. The process of vector quantization consists of encoding and decoding phases.

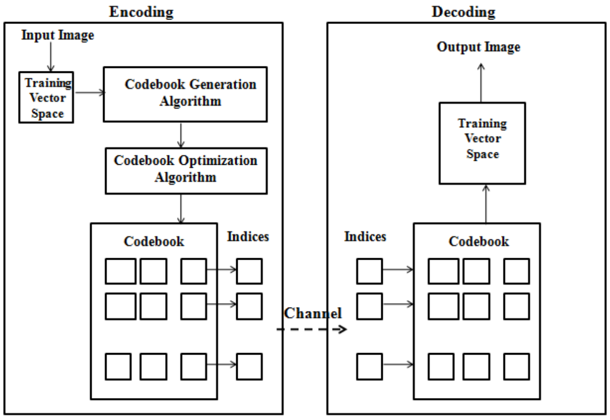

Fig. 1 VQ Image Compression System

A. Encoding Phase

The input image is converted into vectors by dividing the image matrix into 2×2 non overlapping blocks. The RGB components of each pixel are separated and values of each block are written together. This forms the codevector. The collection of all codevectors gives the training vector space of the image. From the training vector space, codebook is generated using different algorithms. The generated codebook is then optimized using the K-means algorithm. An index is created which keeps track of each codevector. The optimized codebook and the index are used to reconstruct the image.

B. Decoding Phase

In the decoding phase, the image is reconstructed back using the optimized codebook and the index. A new training vector space is created in the receiver's side which consists of the codebook values. The correct position is obtained using the index. The image is reconstructed back using this training vector space.

III. CODEBOOK GENERATION AND OPTIMIZATION ALGORITHMS

The various algorithms applied for the generation of codebook are LBG, KPE and KFCG. Apart from these, a new method of Random Selection has also been introduced.

A. LBG Algorithm

The LBG algorithm has been introduced by Linde-Buzo-Gray [9]. First the training vector space is created and the centroid is obtained. The centroid is considered as the first codevector. Now, constant error is added to the codevector and two new vectors are obtained. Fig.2 shows the LBG clustering. v1 and v2 are the new generated codevectors. The Euclidean distance between the training vector space and the vectors v1 and v2 is computed. After computing the Euclidean distance , the training vector which is nearer to v1 is put in one cluster and the training vector which is nearer to v2 is put in another cluster. Two clusters are obtained in the first iteration. In the next iteration, this procedure is repeated for both the clusters.

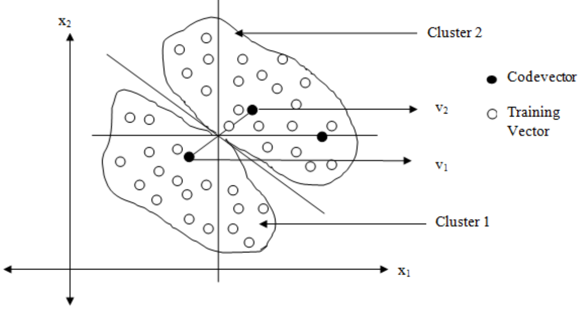

Fig. 2 LBG Clustering

The steps of LBG algorithm are as follows:

1. Generate the training vector space, T of the image which contains M training vectors.
   $T= \{X_1, X_2, X_3 \ldots X_M\}$
   X_i is the training vector which is represented as $X_i= \{x_{i1}, x_{i2}, x_{i3}, \ldots x_{ik}\}$
   where,
   k denotes the dimension.
2. Find Centroid, C of the training vector space by taking the average of each column. This centroid is the first codevector.
   $C= \{C_1, C_2, C_3 \ldots C_M\}$
3. Two new vectors are obtained after adding constant error, E to the codevector.
   C1=C+E and C2=C-E
4. Find the Euclidian distance of the training vector space with these two vectors.
   $D (X_i-C_j) ==\sum_{p=1}^{k}(x_{ip}-c_{jp})^2$
   where,
   X_i is the training vector,
   C_j is the codevector
5. Put the training vector in first cluster if the Euclidian distance between the training vector and the codevector C1 is less else put the training vector in the second cluster.
6. Repeat the steps 2 to 5 for every cluster.
7. Stop when desired codebook size is obtained.

B. KPE Algorithm

Kekre's Proportionate Error (KPE) algorithm [10] uses proportionate error instead of constant error. The training vector space is created and the centroid is obtained. The centroid is the first codevector. This proportionate error is added to the codevector and two new vectors are obtained. The proportionate error is obtained as follows:

- Consider the codevector C and Error Vector E, which is represented as $C = \{c_1, c_2, c_3, \ldots c_k\}$ and
- $E = \{e_1, e_2, e_3, \ldots e_k\}$ respectively.
- Find c_j where $c_j = \min\{c_i / i\}$ and value of i varies from 1 to k.
  j is the index of the vector member whose value is minimum among the other vector members.
- Assign $e_j = 1$
- If $c_i / c_j \leq 10$ then assign $e_i = c_i / c_j$
- Else assign $e_i = 10$ where, $i \neq j$ and value of i varies from 1 to k.

The Euclidean distance between the training vector space and the new vectors is computed. The training vector which is nearer to first vector is put in one cluster and the training vector which is nearer to second vector is put in another cluster. Two clusters are obtained in the first iteration. Thus, the size of the codebook is increased by two. In the next iteration, this procedure is repeated for both the clusters. The codebook size becomes four. This procedure is repeated till the codebook size becomes equal to the size specified by the user.

C. KFCG Algorithm

Kekre's Fast Codebook Generation (KFCG) algorithm [11] does not use Euclidean distance for comparison and also no error is added to get the new vectors. So, the codebook generation is faster as compared to other algorithms . The training vector space is considered as a single cluster initially. The centroid obtained is the codevctor. Fig.3 shows the 1st iteration of KFCG Clustering. C1 is the codevector obtained after centroid calculation. The first member of training vector is compared to the first member of codevector in the first iteration. The training vector is put in first cluster if its first member is less than the first member of codevector. Otherwise the training vector is put in the second cluster. In the second iteration, division of the first cluster is done by comparing second member of the training vector with the second member of codevector. Fig.4 shows 2nd iteration of KFCG Clustering. Again second cluster is divided into two clusters by comparing the second member of the training vector with that of the codevector. This process is repeated till desired codebook size is reached.

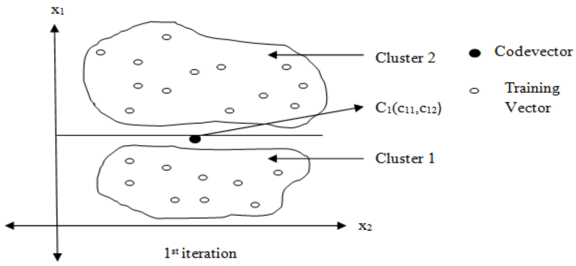

Fig. 3 1st iteration of KFCG Clustering

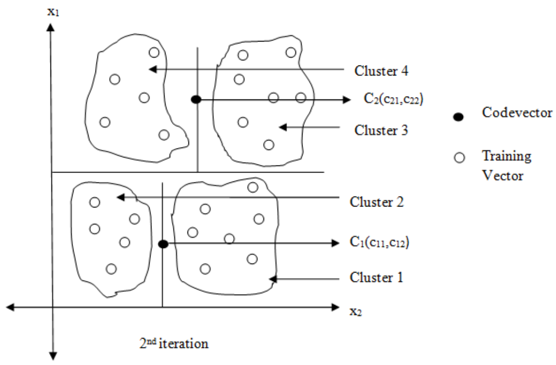

Fig. 4 2nd iteration of KFCG Clustering

A new method of Random Selection has been introduced in this paper for generating the codebook.

D. RANDOM SELECTION METHOD

In this method, codebook of desired size is obtained by random selection of vectors from the training vector space. In other words it can be said that the size of the codebook is dependent on the number of random vectors initially chosen.

The steps are as follows:-

1. Generate the training vector space, T of the image which contains M training vectors.
   $T= \{X_1, X_2, X_3 \ldots X_M\}$
   X_i is the training vector and it is represented as $X_i= \{x_{i1}, x_{i2}, x_{i3}, \ldots x_{ik}\}$
   where,
   k denotes the dimension.
2. Select K random vectors from the training vector space where K denotes the desired codebook size. These initial random vectors serve the purpose of the initial codebook.

The codebook obtained by LBG algorithm, KPE algorithm, KFCG algorithm and Random Selection Method are fed as input to the K-means algorithm for optimization.

E. K-means Algorithm

The K-means algorithm [9] is a clustering algorithm where K denotes the size of the cluster. The basic idea behind this algorithm is to form K clusters and assign each object to one of the K clusters in such a way that the measure of dispersion within the cluster is minimized. The dispersion can be measured using the squared Euclidean distance. This algorithm aims at minimizing the objective function. Here, the mean square error is the objective function. So, the MSE reduces at each iteration and after the codebook is optimized, the MSE reaches a value after which it stops reducing. It converges at the optimal point.

In this algorithm, K-random vectors are selected from the training vector space and call it as codevectors. The squared Euclidean distance of all the training vectors with the selected K vectors are obtained and K clusters are formed.

If the squared Euclidean distance of training vector X_j with i^{th} codevector is minimum then X_j is put in i^{th} cluster. Centroid of each cluster is obtained. The centroids of all clusters obtained in the previous iteration form the set of new codevectors which is the input to K-Means algorithm for the next iteration. The MSE for each of the K clusters is computed and net MSE is obtained. This process is repeated till the net MSE converges. The MSE is calculated using the following formula:-

$$MSE(m)=\frac{1}{Zk}\sum_{r=1}^{Z}(X_r-C_m)^2 \quad (1)$$

where,
m is the cluster number,
Z is the number of vectors in that cluster,
k is the dimension,
X_r is r^{th} training vector,
C_m is the codevector of m^{th} cluster

Instead of selecting K random vectors, the codebooks of size K obtained by LBG, KPE and KFCG, Random Selection Method [12] are used for optimization using the K-means algorithm.

The steps are as follows:-

1. Generate the training vector space, T of the image which contains M training vectors.
   $T= \{X_1, X_2, X_3\ldots X_M\}$
   X_i is the training vector and it is represented as $X_i= \{x_{i1}, x_{i2}, x_{i3},\ldots x_{ik}\}$
   where,
   k denotes the dimension.
2. Generate codebook containing K codevectors using LBG, KPE, KFCG or Random Selection Method where K is the desired codebook size.

3. Find the squared Euclidean distance of all the training vectors with the K codevectors and K clusters are formed.
   $D(X_i-C_j)=\sum_{p=1}^{k}(x_{ip}-c_{jp})^2$
   where,
   X_i is the training vector,
   C_j is the codevector.
4. If the squared Euclidean distance of the X_j with i^{th} codevector is minimum then put X_j in i^{th} cluster.
   If the squared Euclidean distance of X_j with codevectors happens to be minimum for more than one codevector then put X_j in any one of them.
5. Compute the centroid for each cluster.
6. Find MSE for all the K clusters.
7. Calculate net MSE.
8. Replace the initial codevectors by the centroids of each cluster.
9. Repeat steps 3 to 5 till the two successive net MSE values are same.

IV. RESULTS

The techniques are implemented on Intel Core i5-3210M, 2.50 GHz, 4GB RAM machine and Matlab R2011b is used. The algorithms are applied on ten color images each having size of 256×256×3.

Training Images of size 256×256×3 are shown in Fig. 5.

The mean square error and the number of iterations required for optimization of codebook obtained from LBG, KPE, KFCG algorithms and Random Selection Method on codebook size 128, 256, 512 and 1024 are shown in Table I, II, III and IV respectively.

The sample initial and final images of Peacock for LBG, KPE, KFCG and Random Selection Methods with codebook size 256 are shown in Fig. 6.

The Variation of MSE with Number of Iterations for Peacock image for codebook size 256 is shown in Fig. 7.

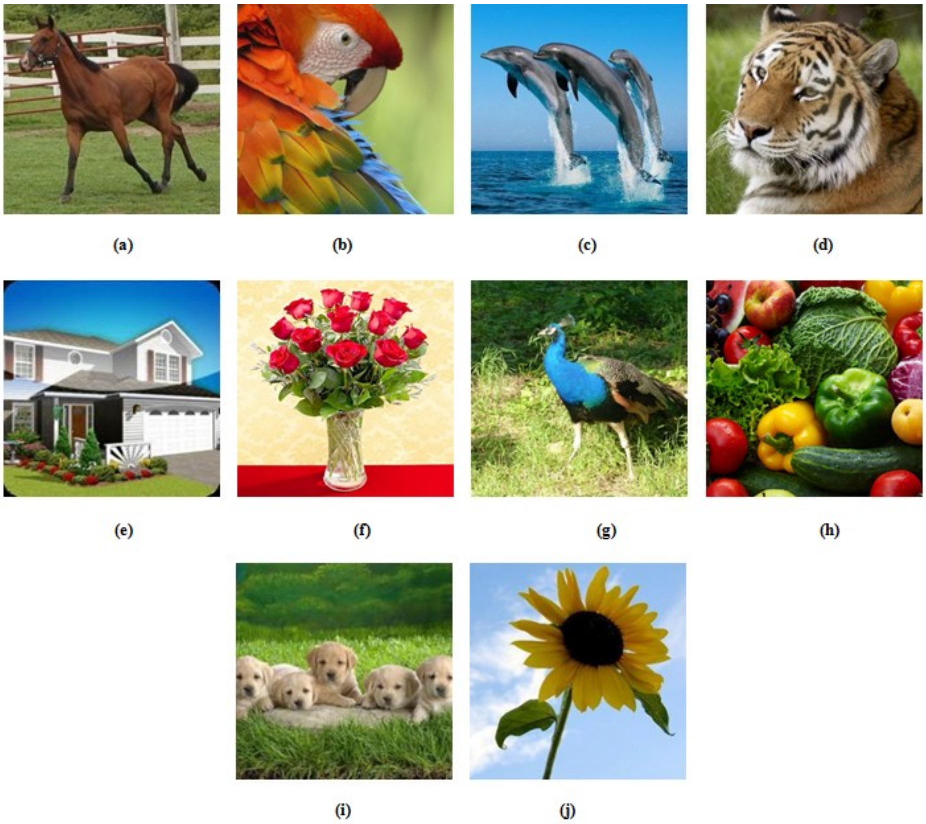

Fig. 5 Training Images (a) Horse (b) Macaw (c) Dolphins (d) Tiger (e) House (f) Roses (g) Peacock (h) Vegetables (i) Puppies (j) Sunflower

Table I. Result of K-means Algorithm on codebook size 128

S. no.	Image	Algorithm	Initial MSE	Final MSE	No. of iterations
1.	Horse	LBG	143.74	64.25	104
		KPE	144.64	63.84	87
		KFCG	194.61	53.03	179
		RANDOM	188.59	69.51	204
2.	Macaw	LBG	259.41	90.19	237
		KPE	225.04	89.35	176
		KFCG	398.26	94.83	409
		RANDOM	273.82	94.06	122
3.	Dolphins	LBG	178.24	86.38	97
		KPE	163.07	87.49	63
		KFCG	244.63	59.69	122
		RANDOM	282.48	112.23	121
4.	Tiger	LBG	247.27	125.18	87
		KPE	252.22	124.29	109
		KFCG	206.11	67.43	90
		RANDOM	256.18	138.38	203
5.	House	LBG	274.22	102.11	112
		KPE	219.31	97.11	76
		KFCG	246.41	74.04	256
		RANDOM	275.16	129.93	192
6.	Roses	LBG	356.11	171.11	116
		KPE	319.22	168.15	118
		KFCG	533.34	127.15	148
		RANDOM	334.85	180.45	158
7.	Sunflower	LBG	115.94	55.13	92
		KPE	119.22	54.57	71
		KFCG	167.79	45.79	278
		RANDOM	168.18	65.98	141

8.	Vegetables	LBG	529.29	224.67	157
		KPE	561.27	228.97	170
		KFCG	704.93	152.15	253
		RANDOM	575.58	233.85	239
9.	Puppies	LBG	141.64	78.73	73
		KPE	140.29	78.45	102
		KFCG	179.49	56.03	141
		RANDOM	186.01	87.12	232
10.	Peacock	LBG	386.11	207.71	98
		KPE	370.94	205.91	112
		KFCG	447.14	156.77	270
		RANDOM	432.55	223.96	275

Table II. Result of K-means Algorithm on codebook size 256

S. no.	Image	Algorithm	Initial MSE	Final MSE	No. of Iterations
1.	Horse	LBG	166.41	56.44	132
		KPE	170.75	56.04	188
		KFCG	179.16	45.99	133
		RANDOM	138.14	56.97	234
2.	Macaw	LBG	285.53	85.45	130
		KPE	232.59	82.46	139
		KFCG	384.51	82.32	412
		RANDOM	494.82	72.31	276
3.	Dolphins	LBG	166.15	74.21	51
		KPE	155.57	76.22	66
		KFCG	220.66	62.81	113
		RANDOM	979.32	118.64	350
4.	Tiger	LBG	245.07	109.83	141
		KPE	235.12	109.55	149
		KFCG	244.54	78.08	156
		RANDOM	264.77	114.96	317
5.	House	LBG	313.73	105.95	174
		KPE	297.59	105.29	187
		KFCG	259.81	85.99	149
		RANDOM	645.79	141.06	330
6.	Roses	LBG	392.79	162.63	135
		KPE	334.15	163.49	143
		KFCG	439.23	131.69	149
		RANDOM	621.99	167.02	136
7.	Sunflower	LBG	119.78	44.72	214
		KPE	100.83	46.99	137
		KFCG	192.81	45.97	283
		RANDOM	536.71	55.04	237
8.	Vegetables	LBG	569.96	212.51	549
		KPE	537.41	215.89	340
		KFCG	520.89	159.79	306
		RANDOM	344.99	189.27	168
9.	Puppies	LBG	148.35	76.56	111
		KPE	152.19	74.56	99
		KFCG	185.71	58.07	175
		RANDOM	659.72	76.53	207
10.	Peacock	LBG	483.52	212.18	97
		KPE	390.01	205.14	161
		KFCG	535.95	165.71	293
		RANDOM	504.49	204.71	180

Table III. Result of K-means Algorithm on codebook size 512

S. no.	Image	Algorithm	Initial MSE	Final MSE	No. of Iterations
1.	Horse	LBG	151.58	49.26	146
		KPE	146.04	49.59	114
		KFCG	176.32	42.54	275
		RANDOM	114.39	44.42	143
2.	Macaw	LBG	223.45	74.14	164
		KPE	211.31	74.63	101
		KFCG	358.09	73.24	252
		RANDOM	370.39	159.51	88

3.	Dolphins	LBG	166.17	64.39	64
		KPE	146.63	67.61	57
		KFCG	244.63	59.69	122
		RANDOM	285.76	106.32	170
4.	Tiger	LBG	227.58	95.91	168
		KPE	220.83	95.61	95
		KFCG	206.11	67.43	90
		RANDOM	287.95	101.91	245
5.	House	LBG	285.06	94.17	161
		KPE	233.02	99.95	128
		KFCG	246.41	74.04	256
		RANDOM	282.61	118.17	191
6.	Roses	LBG	338.45	154.88	196
		KPE	300.37	148.93	148
		KFCG	533.34	127.15	148
		RANDOM	350.39	144.72	116
7.	Sunflower	LBG	120.66	42.31	247
		KPE	89.34	40.43	117
		KFCG	167.79	45.79	278
		RANDOM	364.33	46.89	199
8.	Vegetables	LBG	514.36	205.35	277
		KPE	466.29	201.33	211
		KFCG	704.93	152.15	253
		RANDOM	271.29	158.78	138
9.	Puppies	LBG	137.28	67.92	102
		KPE	139.32	68.12	114
		KFCG	179.49	56.03	141
		RANDOM	642.68	69.26	336
10.	Peacock	LBG	397.88	187.98	117
		KPE	346.79	184.44	120
		KFCG	447.14	156.77	270
		RANDOM	454.69	187.27	151

Table IV. Result of K-means Algorithm on codebook size 1024

S. no.	Image	Algorithm	Initial MSE	Final MSE	No. of Iterations
1.	Horse	LBG	138.01	45.27	82
		KPE	136.51	46.36	92
		KFCG	169.39	39.01	152
		RANDOM	247.12	103.64	54
2.	Macaw	LBG	231.65	66.81	152
		KPE	182.75	68.07	118
		KFCG	374.99		
		RANDOM	862.83	152.07	216
3.	Dolphins	LBG	156.72	57.06	70
		KPE	136.51	60.27	51
		KFCG	325.59	57.04	111
		RANDOM	950.55	124.34	286
4.	Tiger	LBG	212.61	88.11	104
		KPE	212.31	87.18	109
		KFCG	172.34	54.95	66
		RANDOM	232.43	94.57	176
5.	House	LBG	245.54	83.44	172
		KPE	209.25	89.02	222
		KFCG	231.46	81.22	227
		RANDOM	257.95	98.54	250
6.	Roses	LBG	318.04	142.85	167
		KPE	272.03	144.56	153
		KFCG	571.35	122.12	127
		RANDOM	519.31	153.84	173
7.	Sunflower	LBG	111.32	40.67	137
		KPE	78.29	34.87	91
		KFCG	205.23	46.21	132
		RANDOM	581.69	43.76	193
8.	Vegetables	LBG	470.38	195.76	197
		KPE	431.91	194.56	275
		KFCG	507.17	153.23	311
		RANDOM	564.21	363.09	117
9.	Puppies	LBG	136.37	62.91	104
		KPE	137.48	63.81	128

		KFCG	168.21	51.54	205
		RANDOM	607.13	62.62	244
10.	Peacock	LBG	368.88	172.84	148
		KPE	318.03	170.15	123
		KFCG	586.07	153.54	205
		RANDOM	364.42	168.45	128

(a)

(b) (c) (d) (e)

(f) (g) (h) (i)

Fig. 6 Initial and Final Images (a) Original Image (b) Initial image for LBG with MSE=483.52 (c) Final image with MSE=212.18 (d) Initial image for KPE with MSE=390.01 (e) Final image with MSE=205.14 (f) Initial image for KFCG with MSE=535.95 (g) Final image with MSE=165.71 (h) Initial image for Random Selection with MSE=504.49 (i) Final image with MSE=204.71

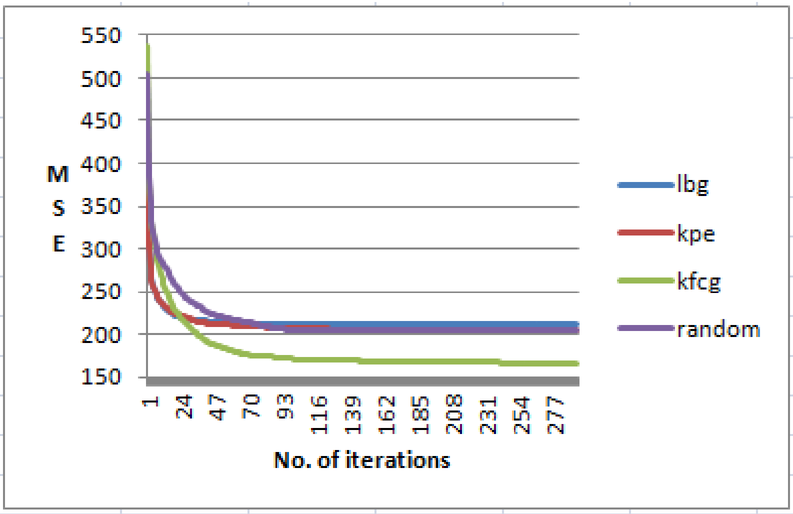

Fig. 7 Variation of MSE with Number of Iterations for Peacock image for codebook size 256

V. CONCLUSION

Codebook generation and optimization can be done using various algorithms. An optimized codebook gives the proper reconstructed image which has less mean square error when compared with the original image. In this paper, codebook is generated using LBG, KPE, KFCG and Random Selection Method. The K-means algorithm is used on the generated codebook for optimization. We have used different images to show the optimization. The codebook size is varied to show the change in mean square error. It is observed that, in most of the cases, the codebook obtained from KFCG algorithm gives less mean square error in comparison to other algorithms. So, when the K-means algorithm is applied to the codebook obtained from KFCG algorithm, the best optimized codebook is obtained .

REFERENCES

[1] R. Navaneethakrishnan, "Study of Image Compression Techniques," International Journal of Scientific & Engineering Research, Vol. 3, No. 7, pp. 1-5, July 2012.
[2] G. Boopathy and S. Arockiasam, "Implementation of Vector Quantization for Image Compression - A Survey," Global Journal of Computer Science and Technology, Vol. 10, No. 3, pp. 22-28, April 2010.
[3] Carlos R.B. Azevedo, Esdras L. Bispo Junior, Tiago A. E. Ferreira, Francisco Madeiro, and Marcelo S. Alencar, "An Evolutionary Approach for Vector Quantization Codebook Optimization," Springer-Verlag Heidelberg, pp. 452-461, 2008.
[4] Pamela C. Cosman, Karen L. Oehler, Eve A. Riskin, and Robert M. Gray, "Using Vector Quantization for Image Processing," In Proc. Of The IEEE, Vol. 81, No. 9, pp. 1326-1341, September 1993.
[5] Tzu-Chuen Lu and Ching-Yun Chang, "A Survey of VQ Codebook Generation," Journal of Information Hiding and Multimedia Signal Processing, Vol. 1, No. 3, pp. 190-203, July 2010.
[6] H.B. Kekre and Tanuja K. Sarode, "Vector Quantized Codebook Optimization using K-Means," International Journal on Computer Science and Engineering, Vol. 1, No.3, pp. 283-290, 2009.
[7] S. Vimala, "Convergence Analysis of Codebook Generation Techniques for Vector Quantization using K-Means Clustering Technique," International Journal of Computer Applications, Vol. 21, No. 8, pp. 16-23, May 2011.
[8] J. B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations", Proceedings of 5-th Berkeley symposium on Mathematical Statistics and Probability", Berkely, University of California Press, vol 1, pp. 281-297, 1967.
[9] 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, January 1980.
[10] H.B.Kekre and Tanuja K. Sarode, "Two-level Vector Quantization Method for Codebook Generation using Kekre's Proportionate Error Algorithm," International Journal of Image Processing, Vol. 4, No. 1, pp. 1-11, 2010.
[11] H.B.Kekre and Tanuja K. Sarode, "Fast Codebook Search Algorithm for Vector Quantization using Sorting Technique," International Conference on Advances in Computing, Communication and Control, pp. 317-325, 2009.
[12] Tanuja K. Sarode and Nabanita Mandal, "K-Means Codebook Optimization using KFCG Clustering Technique," International Journal of Computer Applications, Vol. 78, No. 6, pp. 38-43, September 2013.