CHAPTER 7

NOVEL K-MEANS ALGORITHMS WITH INNOVATIVE STRATEGIES FOR INITIAL SEED SELECTION

7.1 Introduction

K-means is a widely used Vector Quantization technique known for its efficiency and speed. The performance of K-means algorithm strongly depends on the selection of initial seeds. The most common method of initialization is random initialization in which we make a random guess on the initial seeds. Though it is the simplest, it is not reliable and does not always generate good quality codebook. A bad choice of initial seeds will lead to generation of suboptimal codebooks. Many research works have been proposed for enhancing the initialization process [101]-[111]. In this chapter, we propose two new compression schemes that are based on K-Means algorithm in which novel strategies are applied for initial seed selection. The proposed methods are best suited for online web applications that involve massive and rapid image and video storage and transmission.

7.2 Novel K-means

The proposed method is an innovative approach to enhance the performance of K-means algorithm. It adopts divide and conquer strategy by dividing the input data space into many classes and apply K-means clustering.
Novel K-means segments input image into \( K \) clusters and works in two phases. In the first phase, the given image is divided into \( 4 \times 4 \) blocks which are converted into 16 element training vectors. These vectors are divided into \( N \) classes \( C_1, C_2, \ldots, C_N \) based on the mean value of each vector. As the value of a pixel ranges between 0 and 255, the range \( D_i \) of the domain of each class \( C_i \) is calculated as follows:

\[
D_i = \frac{256}{N} \times i
\]  

(7.1)

where \( i \) is the class index with values ranging from 1 to \( N \). A training vector is assigned to a class \( C_i \), if its mean value is less than or equal to the domain of \( C_i \). The motive of this mean based grouping of input data is two-fold. Firstly, it speeds up convergence resulting in reduction of computation time. Secondly, the chance of combining widely differential blocks into the same cluster will be declined and ultimately the average distortion may be further pressed down.

K-means algorithm is performed on each class of vectors with initial seeds. The number of initial seeds \( M_i \) in each class is proportional to its size and they are selected with the right blend of statistical features (variance, mean, median and mode) of the class population.

If high variant vectors outnumber low variant vectors in the class, 50% of the initial seeds are chosen from the vectors having high variance. The remaining seeds are the vectors whose values are mode values of each dimension of the class vectors, in descending order. On the other hand, if low variant vectors dominate the class population (high correlation), the ratio of high variant vectors and high mode vectors would be 1:3. The correlation among data determines the ratio of contribution of high variance and high mode vectors. If the class size is equal to 2, the mean of the two vectors is taken as the representative seed of the class.
The code vectors generated from each class are subjected to pruning based on a minimum (Euclidean) distance threshold “D” with the code vectors of the partially generated codebook. A new code vector $V_{ii}$ of a class $C_i$ will be added to the codebook only if its Euclidean distance with all of the codevectors generated for classes $C_1, \ldots, C_{i-1}$ is greater than a predefined threshold $D$. Thus, pruning of code vectors are done at the construction phase which significantly reduces the codebook size and computation time.

The second phase runs a final round of K-means with the codebook constructed in the first phase as the initial seeds.

### 7.2.1 Algorithm

The input image data is divided into 4×4 blocks and converted into vectors. The number of classes $N$ should be set before the seed selection and pruning phases. The choice of $N$ plays a crucial role in the performance of Novel K-means.

**Seed selection phase**

1. Divide the 512 × 512 input image data into 4 × 4 image blocks and convert the blocks into training vectors (16-vector) of dimension 16.
2. Calculate the range of the domain of each class $D_i$ as given in equation (7.1).
3. Calculate the mean value of each training vector.
4. Assign each training vector to one of the $N$ classes say $C_i$, if its mean value is less than or equal to the domain $D_i$ of the class $C_i$.
5. Calculate $T$ which is equal to the ratio of total number of training vectors and number of initial seeds $K$. 

111
6. Determine the number of initial seeds $M_i$ for each class $C_i$ as follows:

$$M_i = \text{Class-size}[C_i] / T$$

(7.2)

such that $\sum_{i=1}^{N} M_i = K$. Class-size $[C_i]$ is the total number of training vectors in class $C_i$

7. Calculate the variance of each vector and find the number of vectors with variance higher than the mean variance of the class population.

8. Set $m_{1i}$ and $m_{2i}$ to be the number of high variant vectors and mode vectors respectively that form the initial seeds for class $C_i$. If we let $h_{no}$ to be the number of high variant vectors, then

$$if \ h_{no} \geq \text{Class-size}[C_i]$$

$$m_{1i} = M_i / 2$$

$$else$$

$$m_{1i} = M_i / 4$$

$$m_{2i} = M_i - m_{1i}$$

9. Construct the remaining $m_{2i}$ seeds based on the statistical parameters mean, median and mode as follows.

- Take the mean vector of the entire class as the first seed.

- Take the second seed by forming the median vector with the median value of each column in the class.

- Construct mode vectors whose values are the mode value (value with highest frequency of occurrence) of its respective dimension in descending order until we get desired number of seeds ($m_{2i} - 2$). If $\text{mode}_{\text{vector}}[r,c]$ is the $r^{th}$ mode vector, then

$$\text{mode}_{\text{vector}}[r,c] = \text{pixel value}$$

that occupies $r^{th}$ position in descending order of frequency of occurrence (mode) in $c^{th}$ dimension of all the vectors of a class.

10. Perform K-means algorithm on each class with $M_i$ initial seeds.
**Pruning and Construction Phase**

11. Calculate the Euclidean distance between each of the newly generated code vectors of the current class \( C_i \) and the code vectors in the partial codebook constructed from classes \( C_1, \ldots, C_{i-1} \) so far.

12. Add only those code vectors whose Euclidean distance with all the vectors of the partial codebook is greater than a predefined threshold \( D \) to the codebook.

13. Repeat this process for all the classes \( C_i, i=1,\ldots,N \)

14. The final codebook resulting after step 13 is the initial codebook for the entire dataset.

15. Perform final run of K-means on the entire dataset with this final codebook as initial seeds.

**7.2.2 Performance Analysis of Novel K-means**

The performance of the proposed method is measured with several test images of size \( 512 \times 512 \) on a machine with Intel core2 duo processor at 2 GHZ. Our test images are chosen in such a way as to test our algorithm on images having no oscillating patterns (Lena), images with more high frequency components (Barbara), image with both low and high frequency components (Boats). Figure 7.1 shows the histogram of the distribution of pixels for Lena, Barbara and Boats images respectively.
Figure 7.1 Histogram of a) Lena b) Barbara c) Boats

The results obtained for Lena image with $D = 4$ and $K = 256$ are given in Table 7.1. From Table 7.1, it is observed that there is significant drop in computation time with increase in the number of classes without loss in reconstructed image quality and bit rate. Novel K-means on Lena image achieves 70% drop in computation time than K-means with random initialization of seeds. It is observed that it accomplishes abrupt drop in computation time for a particular value of ‘N’ which can be considered as the Break-Even Point (BEP). Experimental results demonstrate that Novel K-means shows optimal performance in terms of computation time, bit rate and PSNR at this BEP and it has also been observed that there is only a small decrease in computation time with increase in value of $N$ above BEP. Figure 7.2 demonstrates the break-even point analysis of test images Lena, Barbara and Boats. Despite the fact that the abrupt drop in computation time occurs at $N = 4$ for images Barbara and Boats, their BEP is 8 as the optimal PSNR and bit rate are achieved only at $N=8$. It is noted that BEP is image-specific and it has been found out to be equal to $c(\log_2K)$ for some constant ‘$c$’ which lies in the range between 0.5 and 1 where $K$ is the number of initial seeds. Several runs of Novel k-means on different test images show that value of $c$ depends on the number of frequently occurring pixel values (hit-no). The value of $c$ is equal to 0.5 for images whose hit-no is less than 200 and 1 for images with hit_no > 200.
Table 7.1 Performance Analysis of BEP for images Lena, Barbara and Boats with \( D = 4 \) and \( K = 256 \)

<table>
<thead>
<tr>
<th>N (No. of Classes)</th>
<th>Lena (BEP = 4)</th>
<th>Barbara (BEP = 8)</th>
<th>Boat (BEP = 8)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Comp. time in sec.</td>
<td>bpp</td>
<td>PSNR</td>
</tr>
<tr>
<td>1 (K-means)</td>
<td>1500</td>
<td>0.125</td>
<td>33.03</td>
</tr>
<tr>
<td>2</td>
<td>1200</td>
<td>0.125</td>
<td>32.90</td>
</tr>
<tr>
<td>4</td>
<td>286</td>
<td>0.122</td>
<td>32.97</td>
</tr>
<tr>
<td>8</td>
<td>200</td>
<td>0.121</td>
<td>32.85</td>
</tr>
<tr>
<td>16</td>
<td>134</td>
<td>0.120</td>
<td>32.84</td>
</tr>
</tbody>
</table>

Figure 7.2 Break-Even Point Analysis for images Lena, Barbara and Boats

Figure 7.3 and Figure 7.4 show the changing pattern of PSNR with the increase in the value of \( N \) for images Lena and Boat respectively. It is evident from Figure 7.3 that PSNR is optimal at \( N = 4 \) (BEP) for Lena and \( N = 8 \) for
Barbara from Figure 7.4. Both figures depict a decline in PSNR with increasing value of $N$ after BEP.

![Figure 7.3 No. of Classes (N) Vs PSNR for Lena image](image)

![Figure 7.4 No. of Classes (N) Vs PSNR for Barbara image](image)

Figure 7.5 shows the visual comparison of the original and reconstructed Lena image using Novel K-means.
Figure 7.5  a) Original Lena image;  b) Reconstructed image using Novel K-means

Table 7.2 shows comparative performance of K-means and Novel K-means with different test images.
Moreover, as the pruning phase prunes redundant code vectors, the size of the final codebook is found to be less than K thereby enhancing compression rate. The threshold ‘D’ used for the pruning process should be chosen so as to achieve better rate-distortion performance. It is learnt from the experiments that this value of $D$ is expected to be in the range 4-6 to get good rate-distortion performance.

Regarding the memory requirements, Novel K-means requires storage for the codebook and code indices as well. If ‘$S$’ is the number of code vectors (16-vector) in the final code book, the size ‘$P$’ of the codebook in bits, is given by

$$ P = S \times 16 \times 8 \text{ bits} $$

(7.3)
Each training vector requires an 8 bit index to find its matching code vector in the decoding process.

Also Novel K-means yields a unique solution for any number of runs whereas K-means with random initialization does not give consistent results. The divide and conquer strategy reduces the computational complexity and eventually time complexity to a great extent without loss of quality and bit rate.

### 7.3 Mode Based K-Means Algorithm with Residual Vector Quantization (MBRVQ)

The next method is an innovative approach that combines the merits of mode based seed selection and residual vector quantization. The motivation of this approach is multi-fold. Firstly, the use of mode parameter for selection of initial seeds exploits correlation among similar blocks for clustering resulting in low bit rate and preserves image quality as well. Second, the use of divide and conquer strategy results in significant reduction of computation time. Third, residual vector quantization phase encodes the residual image, further enhancing the reconstructed image quality.

Mode based K-means involves three processes: Partitioning and Clustering, Pruning and Construction of Master codebook and Residual Vector Quantization. The first two processes are inherited from Novel K-means method. It adopts the divide and conquer strategy by dividing the input data space into classes and apply K-means clustering. Mode based K-means works in three phases. In the first phase, the input vectors are divided into ‘N’ classes based on the mean value of each vector as described in section 7.2.1 The distribution of image data determines the number of classes ‘N’. The range
of the domain of each class is set based on the ratio of distribution of pixels to the number of classes (N).

The range of the domain of each class $D_i$ is calculated as follows:

$$D_i = \frac{(\text{max\_pixel\_value} - \text{min\_pixel\_value})}{N} \times i$$  \hspace{1cm} (7.4)

where $i$ is the Class_index between 1 and N, max\_pixel\_value and min\_pixel\_value are the maximum and minimum pixel values in the input image data respectively.

This mean based grouping of input data accelerates convergence eventually reducing computation time. K-Means algorithm is performed on each class of vectors with initial seeds. The number of initial seeds $M_i$ in each class $C_i$ is proportional to its size and is calculated as

$$M_i = \frac{\text{Class\_size}[C_i]}{T}$$  \hspace{1cm} (7.5)

where, ‘T’ is the ratio of total number of training vectors and number of initial seeds ‘K’ and $\sum M_i = K$, $i = 1,\ldots,N$.

The novelty of this approach is the use of mode for choosing the initial seeds. The initial seeds are the high mode vectors among the class population. High mode vectors are the vectors whose values are mode values of each dimension of the class vectors, in descending order.

If $\text{mode\_vector}[r,c]$ is the $r^{th}$ mode vector, then

$$\text{mode\_vector}[r,c] = \text{pixel value that occupies } r^{th} \text{ position in descending order of frequency of occurrence(mode) in } c^{th} \text{ dimension of all the vectors of a class}.$$  \hspace{1cm} (7.6)

In the second phase, the codevectors generated from each class are subjected to pruning based on a minimum (Euclidean) distance threshold ‘D’ with the code vectors of the partially generated codebook as explained in section 7.2.1. A new code vector $v_{1i}$ of a class $C_i$ will be added to the master
codebook only if its Euclidean distance with all of the code vectors of classes $C_1, \ldots, C_{i-1}$ is greater than a predefined threshold ‘D’. The value of ‘D’ can be used to tune rate-distortion performance. Increase in the value of ‘D’ results in reduction in bit-rate but at the cost of image quality. Experiments have revealed that the optimal value of ‘D’ lies in the range 2-4 to achieve better rate-distortion performance.

The third phase constructs the residual codebook. It computes the residual image by computing the difference between the input vector and the code vector. Next, the residual vectors with the same code index are grouped together which results in ‘K’ groups. One median vector (16-element) is constructed from each group by taking the median value of each column in the group. Next, each median vector (16-element) is converted to 4-element by dividing the 16-element vector into four 4-element blocks and representing each block by its mean value. Thus, the values of 4-element vector are the mean values of the four sub blocks. The resulting ‘K’ 4-element vectors form the residual codebook.

Grouping vectors by their code index provides two advantages. First, input vectors quantized to same code vector are highly correlated which may result in highly correlated residual vector group. This correlation can be utilized to reduce the size of the residual codebook. Second, we don’t require any additional index for residual codebook. One master index is used for both the master codebook and residual codebook as well. This significantly reduces the index overhead, despite the use of two codebooks.
7.3.1 Algorithm

Stage I: Seed generation phase

1. Divide the 512 × 512 input image data into 4 × 4 image blocks and convert the blocks into training vectors (16-vector) of dimension 16.

2. Classify the training vectors into ‘N’ classes based on the mean value of its elements of each vector.

3. Calculate the range of the domain $D_i$ of each class using equation (7.4).

4. Calculate ‘T’ which is equal to the ratio of total number of training vectors and number of initial seeds ‘K’.

5. Determine the number of initial seeds $M_i$ for each class using equation (7.5).

6. Construct mode vectors whose values are the mode value (value with highest frequency of occurrence) of its respective dimension in descending order until we get the desired number of initial seeds $M_i$ using equation (7.6).

7. Perform K-means algorithm on each class with ‘M’ initial seeds.

Stage II: Pruning and Construction Phase

8. Calculate the Euclidean distance between each of the newly generated code vectors of the current class $C_i$ and the code vectors in the partial codebook constructed from classes $C_1, …, C_{i-1}$ so far.

9. Add only those codevectors to the code book whose distance with all of the partial codebook vectors is greater than a predefined threshold ‘D’.

10. Repeat this process for all the classes $C_i$, $i=1, …, N$.

11. The final code book resulting after step 10 is the initial codebook for the entire data set.
12. Perform a final run of K-means on the entire data set with this final codebook as initial seeds.

13. The final code book resulting after step 12 is the master codebook for the entire data set.

**Stage III: Residual Vector Quantization (RVQ) Phase**

14. Compute the residual vector which is the difference between the input vector and its corresponding code vector.

15. Group all residual vectors having the same master code index.

16. Construct the median vector for each group by calculating the median value of each column of the residual vectors belonging to the same group.

17. Convert each median vector (16-element) to 4-element vector by dividing it into four $2 \times 2$ blocks and representing each sub block by its mean value as shown in Figure 7.6.

18. Construct the residual codebook with the 4-element vectors resulting out of step 17.

Figure 7.6 shows the conversion process of 16-element vector into 4-element vector.

![Conversion of 16-element Vector into 4-element Vector](image)
The master code book, residual code book and the code indices form the compressed stream of data.

**Decoding**

1. Determine the code vector from the master code book based on the Master code index of each input vector.

2. Retrieve the residual code vector (4-element) from the residual code book using Master code index.

3. Convert the residual 4-element vector to 16-element vector by replacing the pixel values of the 2x2 sub blocks with their respective mean value.

4. Construct the final code vector by adding or subtracting the corresponding residual values of residual vector (16-vector) from the master code vector.

**7.3.2 Performance Analysis of MBRVQ**

Experiments were carried out using the proposed method MBRVQ on several standard images of size 512 × 512 on a machine with Intel core2 duo processor at 2 GHZ using MATLAB 6.5.

The time taken for code book generation, bit rate in terms of bpp and PSNR values computed for standard images Lena, Barbara, Boat, Peppers and Couple using MBRVQ and K-means algorithm are given in Table 7.3. We note from the Table 7.3 that the time taken for code book generation with MBRVQ has been drastically reduced by 2 – 25 times when compared to K-Means algorithm. Moreover, experimental results reveal that mode based selection of initial seeds in MBRVQ converges faster than Novel K-means for images Barbara, Boat, Peppers and Couple with more oscillating patterns(low and high frequency components). It is further observed that the bit
rate has also improved without significant reduction in the image quality measured in terms of PSNR value and Structural Similarity Index (Q).

In addition, the size of the final codebook is reduced further in the pruning phase and is found to be less than K, thereby enhancing compression rate. It is worth noting that the initial value of K is chosen to be less than 256 to increase bit rate without compromising picture quality by coding the residuals in Residual Vector Quantization phase.

In terms of memory requirements, the proposed method requires storage for the master and the residual codebooks, as well as for storing code indices. As the master codebook consists of a total of K 16-element vectors, the size P in bits of the master codebook is given by \( P = K \times 16 \times 8 \) bits.

However, the residual codebook requires storage only for K 4-element vectors and as the value of each vector element does not exceed 8, the size R in bits of the residual codebook is given by \( R = K \times 4 \times 3 \) bits.

Despite using two codebooks, the memory requirements of the proposed method (P + R) is still less than conventional K-means. Beyond, there is no additional index overhead as one master index is used to retrieve the codevectors from both the master and the residual codebooks. For visual comparison, the original and reconstructed images of Lena by the proposed method are shown in Figure 7.7.
Figure 7.7 a) Original Lena    b) Reconstructed image using K-means    c) Reconstructed image using MBRVQ

Table 7.3 Performance Comparison of MBRVQ on Test Images

<table>
<thead>
<tr>
<th>Image</th>
<th>Method</th>
<th>BEP (N)</th>
<th>Computation time (in sec.)</th>
<th>Bit rate (bpp)</th>
<th>Codebook Size (K)</th>
<th>PSNR</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>K-means</td>
<td>-</td>
<td>1500</td>
<td>0.125</td>
<td>256</td>
<td>33.03</td>
<td>0.681</td>
</tr>
<tr>
<td></td>
<td>MBRVQ</td>
<td>4</td>
<td>350</td>
<td>0.113</td>
<td>208</td>
<td>32.81</td>
<td>0.657</td>
</tr>
<tr>
<td>Barbara</td>
<td>K_means</td>
<td>-</td>
<td>2075</td>
<td>0.125</td>
<td>256</td>
<td>28.14</td>
<td>0.712</td>
</tr>
<tr>
<td></td>
<td>MBRVQ</td>
<td>8</td>
<td>160</td>
<td>0.115</td>
<td>210</td>
<td>27.97</td>
<td>0.693</td>
</tr>
<tr>
<td>Boats</td>
<td>K-means</td>
<td>-</td>
<td>362</td>
<td>0.125</td>
<td>256</td>
<td>31.28</td>
<td>0.645</td>
</tr>
<tr>
<td></td>
<td>MBRVQ</td>
<td>8</td>
<td>150</td>
<td>0.119</td>
<td>214</td>
<td>31.32</td>
<td>0.614</td>
</tr>
<tr>
<td>Peppers</td>
<td>K-means</td>
<td>-</td>
<td>3160</td>
<td>0.125</td>
<td>256</td>
<td>31.40</td>
<td>0.648</td>
</tr>
<tr>
<td></td>
<td>MBRVQ</td>
<td>8</td>
<td>189</td>
<td>0.119</td>
<td>214</td>
<td>31.18</td>
<td>0.632</td>
</tr>
<tr>
<td>Couple</td>
<td>K-means</td>
<td>-</td>
<td>1196</td>
<td>0.125</td>
<td>256</td>
<td>28.89</td>
<td>0.732</td>
</tr>
<tr>
<td></td>
<td>MBRVQ</td>
<td>4</td>
<td>188</td>
<td>0.118</td>
<td>212</td>
<td>28.69</td>
<td>0.714</td>
</tr>
<tr>
<td>Bridge</td>
<td>K-means</td>
<td>-</td>
<td>1368</td>
<td>0.125</td>
<td>256</td>
<td>25.72</td>
<td>0.745</td>
</tr>
<tr>
<td></td>
<td>MBRVQ</td>
<td>8</td>
<td>249</td>
<td>0.116</td>
<td>211</td>
<td>25.45</td>
<td>0.726</td>
</tr>
</tbody>
</table>
For visual comparison, the original and reconstructed images of test images with the proposed methods, Novel K-means and MBRVQ are shown in Figure 7.8 – 7.11.

Figure 7.8 (a) Original Barbara image; Reconstructed image using (b) K-means (c) Novel K-means and (d) MBRVQ

Figure 7.9 (a) Original Boats image; Reconstructed image using (b) K-means (c) Novel K-means and (d) MBRVQ

Figure 7.10 (a) Original Peppers image; Reconstructed image using (b) K-means (c) Novel K-means and (d) MBRVQ
7.4 Conclusions

In this chapter, two methods Novel K-means and Mode-based Residual VQ (MBRVQ) are proposed by using right blend of initial seeds based on statistical features of input data. Novel K-means outperforms conventional K-means with random choice of initial seeds, significantly in terms of computation time with better PSNR and bit rate. The second method MBRVQ is a three phase method that combines K-means and Residual Vector Quantization (RVQ) to further enhance the efficiency of Novel K-means. Mode based K-means shows superior performance than traditional K-means. It obtains a fast solution with better bit rate and comparable PSNR and SSI (Q). This method is an ideal choice for compressing images with high details. Both methods are best suited for time-bound applications of image compression where time complexity plays a crucial role like video conferencing, online search engines of image and multimedia databases.