APPENDIX A

A.1 LBG Design Algorithm

The LBG VQ design algorithm is an iterative algorithm which alternatively solves the above two optimality criteria. The algorithm requires an initial codebook \( C^{(0)} \). This initial codebook is obtained by the \textit{splitting} method. In this method, an initial codevector is set as the average of the entire training sequence. This codevector is then split into two. The iterative algorithm is run with these two vectors as the initial codebook. The final two codevectors are split into four and the process is repeated until the desired number of codevectors is obtained. The algorithm is summarized below.

1. Given \( T \). Fixed \( \varepsilon > 0 \) to be a "small" number.
2. Let \( N = 1 \) and

\[
C^*_1 = \frac{1}{M} \sum_{m=1}^{M} x_m.
\]

Calculate

\[
D_{ave}^* = \frac{1}{MK} \sum_{m=1}^{M} ||x_m - C^*_1||^2.
\]

3. Splitting: For \( i = 1, 2, \ldots, N \), set

\[
C_{i}^{(0)} = (1 + \varepsilon)C^*_i,
C_{N+i}^{(0)} = (1 - \varepsilon)C^*_i.
\]

Set \( N = 2N \)

4. Iteration: Let \( D_{ave}^{(0)} = D_{ave}^* \). Set the iteration index \( i = 0 \).
   i. For \( m = 1, 2, \ldots, M \), find the minimum value of

\[
||x_m - C_n^{(i)}||^2,
\]

over all \( n = 1, 2, \ldots, N \). Let \( n^* \) be the index which achieves the minimum.

Set

\[
Q(x_m) = C_n^{(i)}.
\]
ii. For \( n = 1,2 \ldots, N \), update the codevector

\[
c_n^{(i+1)} = \frac{\sum_{Q(x_m) = c_n^{(i)}} x_m}{\sum_{Q(x_m) = c_n^{(i)}} 1}
\] 

\( \ldots \ldots (A-5) \)

iii. Set \( i = i + 1 \).

iv. Calculate

\[
D_{ave}^{(i)} = \frac{1}{Mk} \sum_{m=1}^{M} ||x_m - Q(x_m)||^2.
\] 

\( \ldots \ldots (A-6) \)

v. If \( (D_{ave}^{(i-1)} - D_{ave}^{(i)}) / D_{ave}^{(i-1)} > \epsilon \) go back to Step (i).

vi. Set \( D_{ave}^{*} = D_{ave}^{(i)} \). For \( n = 1,2 \ldots, N \), set

\[
c_n^{*} = c_n^{(i)}
\]

as the final codevectors.

5. Repeat Steps 3 and 4 until the desired number of codevectors is obtained.

A.2 Back Propagation algorithm

As discussed earlier it’s a multilayer feed forward network with backward error (gradient) propagation. The function used is sigmoid. It learns mapping from a set of input pattern (extracted features) to a set of output pattern (Classes). It likely gets stuck in local minimum. (drawback)

- Weight Initialization

Set all weights to small random numbers.

- Calculation of activation.

The activation level of an input is determined by the instance presented to the network. The activation level \( Y_j \) of hidden and output unit is determined by

\[
Y_j = F(W_jX_i - \theta_j)
\] 

\( \ldots \ldots \ldots (A-7) \)

Where \( W_{ji} \) is the weight from input \( X_i \)
Weight Training

Start at the output units and work backward to the hidden layers recursively
\[ \Delta W_{ji} = \eta \delta_j x_i. \] \hspace{1cm} (A-9)

The weight change is computed by
\[ W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}. \] \hspace{1cm} (A-8)

Learning rate \( \eta \)

Error gradient \( \delta \)

Convergence is sometimes faster by adding a momentum term
\[ W_{ji}(t+1) = W_{ji} + \eta \delta_j x_i + \alpha [W_{ji}(t) - W_{ji}(t-1)]. \] \hspace{1cm} (A-10)

The error gradient is given by

For the output units
\[ \delta_j = O_j (1 - O_j) (T_j - O_j). \] \hspace{1cm} (A-11)

For the hidden units
\[ \delta_j = O_j (1 - O_j) \sum_k \delta_k W_{kj}. \] \hspace{1cm} (A-12)

Where \( \delta_k \) is the error gradient at unit \( k \) to which the connection points from hidden unit \( j \).

Repeat iterations until convergence in terms of the selected error criterion.
APPENDIX B

B.1 Original Gray Scale test images used for evaluation of different Vector quantizers and Not presented in Chapter IV

Fingerprint

Woman 1 (Tiffany)

Woman 2

Zelda

Bird

Peppers 1
Girlface  Masuda

Cup  Bridge

X-Ray  DIP
B. 2 Original Medical test images (not presented in Chapter IV)

CT abdomen

CT angiogram
B.3 Original color test images used for evaluation of performance of various codebooks (not presented in chapter V)

Pepper 1

Masuda

Sail boat

Parrots

House

Colorblocks 1
Koyana dam

Flowers
Start

Divide Lena image into 4x4 blocks, i=1

Give Ith blocks to SOFM network with α = 0.005, No.of epochs =100

Record weight Vector as code vector

No. of blocks = 1024

Y

Stop

N

i=i+4

(a)

Start

Divide Test image into 4x4 blocks

Take block and compare with the codevectors and store minimum distance in array

No. of codevectors = 1024

Y

Find minimum distance and store its address

No. of blocks = 4096

N

Map the image

Stop

(b)

Fig. B.1 (a) Flowchart for training of (design) of Vector Quantizer using Lena image and Enhanced VQ using own developed image.(Replace Lena by own image) . (b) Flowchart for testing (compression and decompression) of image using designed VQ.
Start

Divide Lena, Peppers, CT scan, gray level etc. image into 4x4 blocks. i = 1

Calculate Mean, Std. Deviation and Apply canny edge operator

If mean < 10 and std dev < 5
  Initialize weights to "random"

If mean < 100 and std dev < 20
  Initialize weights to "zero"

Initialize weights to "midpoint"

Give I th blocks to SOFM network with \( \alpha = 0.005 \), No. of epochs = 100 for mean < 10 and > 100 otherwise of epochs = 10

Calculate error between original and trained and store in codebook after 512

No. of blocks = 1024

Stop

Fig. B.2 Flowchart of design process of Generic with error codebook. During reconstruction the error information is searched and added to the codevector.