4 VECTOR QUANTIZATION

4.1 OVERVIEW
This chapter gives brief introduction of quantization technique known as vector quantization. In this chapter basic principle of vector quantization is explained. Various vector quantization types such as gain shape VQ, classified VQ, interpolative VQ etc. are also described.
All spatial domain VQ techniques are applicable to transform VQ such as wavelet based VQ. However wavelet based techniques also take advantage of wavelet transform properties.
This chapter gives details of experimentation done related to wavelet VQ. Basic wavelet based VQ is evaluated.
Three new methods of VQ are proposed here, they are triangular inequality based classified VQ, modified interpolative VQ and priority based VQ. Methods are explained in this chapter.

4.2 Motivation
From the literature survey, it is clear that different VQ techniques are tried by researchers. VQ outperform scalar quantization. VQ in wavelet domain gives still more compression than spatial domain. But speed of the VQ encoding is the main problem, which limits use of VQ. Fast search techniques are implemented, but mostly in spatial domain.
There seems to be less work for fast search in wavelet domain. Fast search algorithms can be grouped into three categories: partial distortion elimination (PDE), triangular inequality elimination (TIE), and mean-distance-ordered search (MOS).
Mean ordered search technique is not suitable for wavelet domain because wavelet coefficients take positive as well as negative values. Hence mean can not be used as codevector matching criterion.

Aim of this chapter is to develop fast search VQ technique in wavelet domain, which will make use of wavelet properties.
4.3 Introduction

In 1980, Linde, Buzo and Gray (LBG) proposed a VQ design algorithm. Vector quantization (VQ) is a lossy data compression method based on the principle of block coding. It is similar to scalar quantization technique. In scalar quantization single data sample is approximated to nearest quantization level. However in vector quantization group of data samples or 'vector' is approximated using codebook. Codebook is generated by selected vectors from training sequence i.e. huge database of images.

The number of data samples in group decides dimension of vector. These vectors are approximated by same dimensional vectors in codebook.

A vector quantizer \( Q \) of dimension \( K \) and size \( N \), is mapping from a vector in \( K \) dimensional Euclidean space \( \mathbb{R}^k \) into finite set \( C \) containing \( N \) output or reproduction points called codevectors. Thus \( Q : \mathbb{R}^k \rightarrow C \)

where \( C = \{ y_1, y_2, y_3, \ldots, y_N \} \) and \( y_i \in \mathbb{R}^k \) for each \( i \in J = \{ 1, 2, 3, \ldots, N \} \) the set \( C \) is called the codebook. Each coedvector has the size of \( K \). Rate of vector quantizer equals to number of bits per vector component used to represent the input vector.

Associated with every \( N \) point vector quantizer is a partition of \( \mathbb{R}^k \) into \( N \) regions or cells for \( i \in J \). the \( i^{th} \) cell is defined by

\[ R_i = \{ x \in \mathbb{R}^k : Q(x) = y_i \ \cup \ R_i = \mathbb{R}^k \text{ and } R_i \cap R_j = 0 \text{ for } i \neq j \]

A vector quantizer is called regular if each cell \( R_i \) is convex set and for each \( i \), \( y_i \in R_i \).

![Figure 4.1 Partitioning of space](image-url)
Vector quantizer can be decomposed into two parts i.e. encoder and decoder as shown in figure 3.2. The encoder \( e \) is the mapping from \( \mathbb{R}^k \rightarrow J \) and the decoder \( D \) maps the index set \( J \) into the reproduction set \( C \).

\[
e : \mathbb{R}^k \rightarrow J \quad \text{and} \quad D : J \rightarrow \mathbb{R}^k
\]

![Figure 4.2- Block Diagram of Vector quantization](image)

Data (Image) \rightarrow Group into vectors \rightarrow Find closest match \rightarrow codebook \rightarrow index \rightarrow X

![Figure 4.3 - Vector quantization encoding](image)

Overall procedure can be explained with the help of figure 4.2 & 4.3

We have codebook derived from training set. For every input vector we find best match vector from codebook. We send index of the matched codevector to the decoder. Decoder has same codebook. From index decoder can know the best match vector. At the decoder side we approximate input vector by best-matched vector.

Larger the codebook we get closer match to the input vector hence distortion is reduced. But number of bits needed for index value representation increase. This reduces compression ratio. At the same, search time and storage space for codebook increases.

**Optimality Conditions For VQ**

The principal goal in design of vector quantizers is to find a codebook, specifying the decoder, and a partition or encoding rule, specifying the encoder that will maximize an
overall measure of performance considering the entire sequence of vectors to be encoded.

Statistical average of the distortion for a vector quantizer $Q(.)$ can be expressed as

$$D = E_d(x, Q(x)) = \int d(x, Q(x)) f_x(x) dx \quad (4.1)$$

Where $f_x(x)$ is the probability distribution function of input vector $X$ and integration is multiple integration over $K$ dimensional space.

When input vector has discrete distribution distortion can be expressed as

$$D = E_d(x, Q(x)) = \sum_{x_i} d(x_i, Q(x_i)) p_x(x_i) \quad (4.2)$$

Where $\{x_i\}$ are the values of $X$ that have nonzero probability.

As encoder is completely specified by the partition of $R^k$ into the cells $R_1, R_2, \ldots, R_N$ and the decoder is completely specified by the codebook $C = \{y_1, y_2, \ldots, y_N\}$.

The optimality conditions explicitly determine the optimal partition for a given codebook and optimal codebook for a given partition. For a given codebook an optimal partition is one satisfying nearest neighbor condition.

**Nearest Neighbor condition**

For a given set of output levels $C$ the optimal partition cells satisfy

$$R_i \subset \{x : d(x, y_i) \leq d(x, y_j) \text{ for all } j\} \quad (4.3)$$

That is

$$Q(x) = y_i \text{ only if } d(x, y_i) \leq d(x, y_j) \text{ for all } j \quad (4.4)$$

Thus given the decoder, the encoder is minimum distortion or nearest neighbor mapping and hence
\[ d(x, Q(x)) = \min d(x, y_i), y_i \in C \]  \hspace{1cm} (4.5)

**Centroid condition- for a optimal codebook**

for a given partition \( \{ R_i, i = 1 \ldots N \} \) the optimal code vectors satisfy

\[ y_i = \text{centroid}(R_i) \text{, where} \]

\[ \text{centroid } R_i = \frac{1}{\|R_i\|} \sum_{j=1}^{N} x_j \]  \hspace{1cm} (4.6)

**Measuring Vector Quantizer Performance**

Performance of vector quantizer can be measured in terms of cost function

\[ d = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} d(x, x) \text{ where } d(x, x) \]  \hspace{1cm} (4.7)

is the cost associated with quantizing input vector \( x \).

Most convenient cost function and Euclidean distance between the two vectors

\[ d(x, x) = \|x - \hat{x}\|^2 = \sum_{i=1}^{N} (x_i - \hat{x}_i)^2 \]  \hspace{1cm} (4.8)

**4.3.1 Nearest Neighbor Quantizer**

Important class of vector quantizer is Vernoi or nearest neighbor vector quantizer. Partition of the quantizer is completely determined by the codebook and distortion measure. It requires no geometrical description of the cells. For the distortion measure square error

\[ d(x, y) = \|x - y\|^2 = \sum_{i=1}^{N} (x_i - y_i)^2 \]  \hspace{1cm} (4.9)
\[ R_i = \left\{ x : d(x_i, y_i) < d(x_i, y_j) \text{ all } j \neq i \right\} \] 

(4.10)

### 4.3.2 Methods Of Codebook Generation

codebook design is very important in vector quantization. Well designed codebook reduces overall distortion. Initial codebook can be generated by selecting code vectors at random from training set. splitting or pruning technique can also be used for this purpose. Initial codebook generated can be improved by using generalized Lloyd algorithm, which satisfies optimality condition. This algorithm improves codebook iteratively.

**Lloyd iteration for codebook improvement**

Given a codebook \( C_m = y \), partition the training set into cluster sets \( R_i \) using the nearest neighbor condition.

\[ R_i = \left\{ x \in J \ d(x, y_i) < d(x, y_j) \text{ all } j \neq i \right\} \] 

(4.11)

Using the centroid condition compute the centroids for the cluster sets just found to obtain the new codebook \( C_{m+1} = \text{cent}(R_i) \) based on Lloyd iteration.

**4.3.3 Generalized Lloyd algorithm is**

Step1 - begin with an initial codebook \( C_1 \), set \( m = 1 \)

Step2- given the codebook \( C_m \) perform the Lloyd iteration to generated improved codebook \( C_{m+1} \)

Step3 – compute the average distortion for \( C_{m+1} \). If it is changed by small enough amount since the last iteration, stop otherwise go to step2
4.4 Constrained Vector Quantization

Vector quantizer performance is far more superior compared to scalar quantizer in terms of compression ratio. But encoding task is more complex. VQ takes long time to find closest match codevector from codebook. To reduce the search complexity, some constraints are applied on the codebook, i.e., the codeword cannot have arbitrary locations as points in K-dimensional space but are distributed in a restricted manner that allows easier search for the nearest neighbor.

4.4.1 Types of constrained search techniques

Lattice VQ: Codebook is composed of all integral combinations of a set of linearly independent vectors which span the space. In two-dimensional lattice, a set of points with regular arrangement in the plane \( \mathcal{L} = \{ t \in \mathbb{R}^2 \mid t = Am \} \) for all integer pairs \( m = (m_1, m_2) \), where \( A \) is a nonsingular (2 x 2) matrix called the generator matrix of the codebook.
lattice. ‘m’ is an integer vector. This concept can be generalized to N dimension. For each cell the region of support of lattice VQ reproduction vector is midpoint or centric of the cell.

**Gain shape VQ** - input vectors are first encoded into energy shape vector by maximum correlation search and then optimum gain for the given shape is found by scalar quantization.

**Tree structured VQ (TSVQ)** - in TSVQ the codeword is generated by a sequence of binary minimum distortion decision, comparing the input vectors to stored reproduction vectors (codevectors) at each available node. The encoder produces binary symbols to represent its sequence of decisions from the root node of tree through the terminal node. Path map is the final codeword.

**Multistage vector quantization** - multistage VQ divides the encoding task into several stages. First stage uses small codebook and finds approximate vector. Second stage operates on the error vector between original vector and the quantized first stage output. Quantized error vector provides refinement to the first approximation. At the decoder the reproduction vector produced by the first and second will be added together.

**Finite state vector quantization** - it has multiple codebooks. Codebook to be referred is decided by next state rule.

Trellis coded vector quantization it is having supercodebook, which is partitioned into collection of subcodebooks. Encoder has a collection of possible states and allowed transition between these states. Each of the transition between states corresponds to one of the subcodebooks. Codeword is made of two parts. First gives index, which gives allowable transition being taken. While second part gives index of the codeword. Some of the constrained techniques which are used for wavelet coefficient VQ are explained in more details.

### 4.4.2 Classified VQ

The input Xn is subjected to classifier, which generates an index from 1 to m, which identifies the sub codebook to search for the nearest neighbor. Codeword corresponding to Xn is combination of index in specifying the codebook and index specifying selected word. Codebook is partitioned into unequal sized subcodebooks. Classifier will select particular subcodebook based on predefined criterion. The classifier generates an index, which identifies the subcodebook to search for nearest neighbor. The codeword
consists of the index ‘i’ specifying which of the m codebook is selected and \( \log_2(c_j) \) bits specifying the selected word in the codebook. The classifier in a CVQ encoder can extract one or more features from the input vector such as the mean, energy, difference between minimum and maximum or other statistics such as zero crossing counts etc.

Classified VQ

![Diagram of Classified VQ](image)

**Figure 4.5 – Classified VQ**

### 4.4.3 Shape Gain VQ

It is product code technique that decomposes the problem into that of coding a scalar and a vector. Scalar quantity to be coded is root mean square value or energy of the vector. This is known as gain and serves as normalizing scale factor. The normalized input vector is the shape vector.

Gain

\[
g = \|X\| = \sqrt{\sum_{i=1}^{N} X_i^2}
\]  \hspace{1cm} (4.12)

\[
S = \frac{X}{g} \text{ so that } \|s\| = 1
\]  \hspace{1cm} (4.13)

Distortion in vector encoding
\[ d(x, gs) = \|X - gs\|^2 \]  

(4.14)

\[ d(x, gs) = \|X\|^2 + g^2 - 2g(x, s) \]  

(4.15)

**Shape gain VQ encoder**

![Shape gain VQ encoder diagram]

**Shape gain decoder**

![Shape gain decoder diagram]

Figure 4.6- Shape gain VQ encoder and decoder
4.4.4 Interpolative VQ

In a signal if large number of samples are interdependent high dimensional vector can be formed for better efficiency. Encoding such high dimensional vector at higher rates increase search complexity. So alternative approach can be used, i.e. from high dimensional supervector lower dimensional feature vector can be extracted. The feature vector is coded by suitable technique. Quantized feature vector, either by itself or in conjunction with other information can generate an appropriate description of the supervector. Feature vector or subvector can be extracted from supervector by using subsampler. Subsampled vector is quantized and encoded. While at the decoder supervector can be reconstructed from quantized vector. This scheme is known as interpolative VQ.

Interpolative VQ

![Interpolative VQ Diagram](image)

Figure 4.7- Interpolative VQ
Interpolation method can be linear or non linear. In linear method supervector can be predicted using equation

\[ Y = AX \]  

(4.16)

Where \( X \) is the given \( N \) dimensional vector and \( A \) from which we want to predict \( k \) dimensional vector \( Y \). \( A \) is \( K \times N \) dimensional matrix such that

\[ A \ast R_x = E(Y \ast X) \]  

(4.17)

Where \( R_x \) is autocorrelation matrix

In Nonlinear Interpolative VQ – \( K \) dimensional feature vector is quantized using encoder codebook \( C \). Index of nearest codeword is send to the decoder. Decoder codebook is different from encoder. From index ‘I’ decoder gives \( n \) dimensional reconstructed vector. Decoder codebook can be found from training set. Centroids of all those input vectors which are partitioned into \( i^{th} \) region is \( i_{th} \) codeword in \( C^* \).
\[ C_i^* = \frac{1}{N_i} \sum_{j:x_j} R \cdot x_j \]  

(4.18)

### 4.5 Transform VQ

Instead of directly applying a signal vector \( X \) as input to a vector quantizer an orthogonal linear transformation \( T \), as used in transform condition can be first applied to the input vector. The transformed vector \( Y \) then can be vector quantized and the quantizer output \( Y^* \) can be inverse transformed to yield quantized approximation \( X^* \) to the original input.

![Figure 4.9- Transform VQ](image)

Average distortion is same as the case of direct VQ without using a transformation. Advantage of transform VQ is that linear transformation gives compact representation of image. Therefore substantial fraction of the components of the transformed signal vectors are very close to zero and can be neglected entirely.

#### 4.5.1 Wavelet transform and VQ

VQ techniques can be applied to transformed coefficients. Vector formation can be done by two methods

1. Same level cross band vectors - which groups the pixel at same level and same location.
2. Difference level cross-band vectors which groups the pixels at different levels but same location.
4.6 Experimentation

4.6.1 Wavelet based VQ.

Five images cameraman, bridge, bird, peppers, Zelda are selected as training images. These images are not included in test set images. N level decomposition is done using Biorthogonal wavelet filter. N is varied from 2 to 4. Using Lloyd’s algorithm codebook was generated from training images. Codebook size is varied from 128 to 4096. To form vectors wavelet coefficients are grouped either by same orientation all levels or same level all orientation as shown in figure in (4.10 and 4.11). After VQ encoding, codebook index were entropy coded using arithmetic coding. Effect on performance due to decomposition level variation, codebook size variation and vector formation method was studied.

- There exists correlation between coefficients at different level and different orientation. Grouping coefficient of either same level different
orientation or different level same orientation can be done. Performance evaluation of these two techniques is done. Image is decomposed using bior6.8 filter. For ‘same orientation cross band method’, 1 coefficient from lowest (coarsest), 4 coefficients from next higher level and 16 coefficients from highest band are grouped to form 21 dimensional vectors.

- LL3 coefficients are scalar quantized using 8 bit scalar quantizer.
- Minimum compression ratio is limited by bits required to code LL3 band. With 8 bit scalar quantizer this lower limit becomes .125 bits/pixel.
- To reduce compression ratio below 0.125 b/pixel four level wavelet decomposition was used. This gives lower bound of compression to compression ratio .03bit/pixel
- To very compression ratio codebook size can be varied. Codebook size is varied from 128 to 4096
- It is also not practical to compression ratio to 1bit/pixel or higher. Because this requires codebook size of $2^{18}$, so encoding procedure will take very large time. We can use two level decomposition for lower compression ratio.

### 4.6.2 Experiment 1- Vector quantization in Wavelet domain same orientation all levels crossband

<table>
<thead>
<tr>
<th>Image</th>
<th>b/p</th>
<th>PSNR</th>
<th>B/pixel PSNR</th>
<th>b/p</th>
<th>PSNR</th>
<th>b/p</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>.33</td>
<td>33.97</td>
<td>.39 34.04</td>
<td>.43</td>
<td>34.92</td>
<td>.48</td>
<td>35.04</td>
</tr>
<tr>
<td>Boats</td>
<td>.33</td>
<td>30.98</td>
<td>.41 31.25</td>
<td>.44</td>
<td>32.56</td>
<td>.49</td>
<td>33.24</td>
</tr>
<tr>
<td>Peppers</td>
<td>.34</td>
<td>32.40</td>
<td>.42 33.66</td>
<td>.43</td>
<td>34.11</td>
<td>.48</td>
<td>35.68</td>
</tr>
<tr>
<td>Zelda</td>
<td>.32</td>
<td>36.51</td>
<td>.40 37.90</td>
<td>.42</td>
<td>38.15</td>
<td>.47</td>
<td>39.14</td>
</tr>
<tr>
<td>Mandrill</td>
<td>.35</td>
<td>25.58</td>
<td>.42 26.70</td>
<td>.45</td>
<td>28.86</td>
<td>.51</td>
<td>29.75</td>
</tr>
<tr>
<td>Barbara</td>
<td>.35</td>
<td>27.93</td>
<td>.42 29.04</td>
<td>.44</td>
<td>29.91</td>
<td>.50</td>
<td>30.88</td>
</tr>
<tr>
<td>Goldhill</td>
<td>.35</td>
<td>31.68</td>
<td>.41 32.87</td>
<td>.44</td>
<td>33.18</td>
<td>.49</td>
<td>33.85</td>
</tr>
</tbody>
</table>

Table 4.1- Same orientation all levels VQ – 3 decomposition levels
<table>
<thead>
<tr>
<th>Image</th>
<th>B/p</th>
<th>PSNR</th>
<th>b/pixel PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>1.31</td>
<td>39.85</td>
<td>1.5</td>
</tr>
<tr>
<td>Boats</td>
<td>1.32</td>
<td>34.17</td>
<td>1.52</td>
</tr>
<tr>
<td>Testpat1</td>
<td>1.29</td>
<td>27.18</td>
<td>1.5</td>
</tr>
<tr>
<td>Testpat2</td>
<td>1.29</td>
<td>18.26</td>
<td>1.49</td>
</tr>
<tr>
<td>Barb</td>
<td>1.33</td>
<td>32.48</td>
<td>1.54</td>
</tr>
<tr>
<td>Mandrill</td>
<td>1.35</td>
<td>27.57</td>
<td>1.54</td>
</tr>
<tr>
<td>Goldhill</td>
<td>1.34</td>
<td>33.94</td>
<td>1.52</td>
</tr>
<tr>
<td>Hurrican</td>
<td>1.30</td>
<td>38.80</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Table 4.2 - Same orientation all levels VQ - 2 decomposition levels

4.6.3 Experiment 2 - Same level all orientation crossband VQ

- For ‘same level all orientation’ method, at lowest frequency level 1 coefficient from every orientation are grouped. At next higher-level 2x 2 coefficients from every orientation are taken this makes 12 dimensional vector. Similarly 4 X 4 coefficients at next higher level from each orientation make vector size 48.

- 3 different codebooks are used for three different levels. Codebook size i.e. number of codevectors are kept same for all three levels. This makes compression ratio more at higher level but overall compression ratio same as same orientation cross band. To vary compression ratio codebook size was varied. For this method codebook size can be kept different for different level of decomposition.

- Different combination of codebook size were tried

<table>
<thead>
<tr>
<th>Image</th>
<th>B/p</th>
<th>PSNR</th>
<th>B/pixel PSNR</th>
<th>b/p</th>
<th>PSNR</th>
<th>b/p</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>.39</td>
<td>33.23</td>
<td>.40</td>
<td>33.70</td>
<td>.43</td>
<td>34.45</td>
<td>.58</td>
</tr>
<tr>
<td>Boats</td>
<td>.40</td>
<td>30.50</td>
<td>.41</td>
<td>31.15</td>
<td>.45</td>
<td>32.0</td>
<td>.60</td>
</tr>
<tr>
<td>Testpat2</td>
<td>.39</td>
<td>14.99</td>
<td>.41</td>
<td>15.39</td>
<td>.43</td>
<td>16.82</td>
<td>.45</td>
</tr>
<tr>
<td>Barb</td>
<td>.43</td>
<td>27.92</td>
<td>.45</td>
<td>28.63</td>
<td>.48</td>
<td>29.95</td>
<td>.51</td>
</tr>
</tbody>
</table>
Table 4.3 - All orientation same level VQ- 3 level decomposition

<table>
<thead>
<tr>
<th>Image</th>
<th>B/p PSNR</th>
<th>B/pixel PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>1.31</td>
<td>1.5</td>
</tr>
<tr>
<td>Boats</td>
<td>1.32</td>
<td>1.52</td>
</tr>
<tr>
<td>Testpat1</td>
<td>1.29</td>
<td>1.5</td>
</tr>
<tr>
<td>Testpat2</td>
<td>1.29</td>
<td>1.49</td>
</tr>
<tr>
<td>Barb</td>
<td>1.33</td>
<td>1.54</td>
</tr>
<tr>
<td>Mandrill</td>
<td>1.35</td>
<td>1.54</td>
</tr>
<tr>
<td>Goldhill</td>
<td>1.34</td>
<td>1.52</td>
</tr>
<tr>
<td>Hurrican1</td>
<td>1.30</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Table 4.4 - All orientation same level using 2 level decomposition

![Graph](image)

Figure 4.12 – Comparison of crossband VQ methods
4.6.4 Conclusion and discussions of

1. Two cross band vector grouping methods are studied i.e. ‘same level all orientation’ and ‘same orientation all levels’. For ‘same level all orientation’ we used separate codebook for every level. All codebooks are of same size. For same orientation method only one codebook is used. Codebook size are varied from 256 to 1024 for both the methods.

2. With both grouping scheme more compression is achieved compared to scalar quantization.

3. Compression ratio or bits/pixel is function of vector dimension and size of codebook. Increasing size of the codebook decreases compression ratio. Increase in vector size increases compression. Bits/pixel is proportional to codebook size and inversely proportional to vector size. Thus to vary compression ratio we can either change vector size or codebook size. For higher bit rate we can use large codebook. But this increases search time and also codebook storage. This imposes upper limit on codebook size. We can reduce then vector dimension. Similarly for very low bits/pixel it is more suitable to increase vector dimension instead of reducing codebook size to very small value as smaller codebook also affects quality.

4. Compression results of VQ implementation are better than JPEG but inferior to JPEG 2000 up-to bit rate 1.5. Above 1.5 bit rate performance is better than JPEG 2000.

5. Out of this ‘same level all orientation’ method gives more compression than ‘same orientation all levels’.

4.6.5 Classified VQ based on triangle inequality principle.

To find the best-matched codevector, we need a matching criterion and a codebook search algorithm. The most popular matching criterion is the Euclidean distance. In order to find the best-matched codevector, the basic VQ employ the full search algorithm (FSA), which calculates the Euclidean distance between the input vector and every codevector in the codebook. The comparison of an input vector with codevectors is generally referred to as a codebook search problem.

Full search technique compares input vector with every codevector to find best match.
This takes very long time, especially for large codebook. To increase the speed of
operation, fast search techniques are used. One such technique uses triangle inequality
principle.

**Codebook Search Based On Mathematical Triangle Inequality**

The codebook search based on triangle inequality elimination (TIE) with a single
control vector (control point).

Let \( X = \{ x(i,j) ; i, j = 1, 2, \ldots, 2^n \} \)

\( Y = \{ y(i,j) ; i, j = 1, 2, \ldots, 2^n \} \)

\( C = \{ c(i,j) ; i, j = 1, 2, \ldots, 2^n \} \)

Let in a dimensional Euclidean space, \( x(i,j) \) represents the input image vector (block)
to be encoded, \( y(i,j) \) represents a codevector in the VQ codebook, and \( c(i,j) \) is a control
vector.

The squared Euclidean distance between \( x(i,j) \) and is defined as

\[
\| x - y \|^2 = \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} (x(i,j) - y(i,j))^2.
\]

(4.19)

For the three vectors, the mathematical triangle inequality states that

\[
|d(x, c) - d(y, c)| \leq d(x, y)
\]

(4.20)

to illustrate the use of triangle inequality in the codebook search problem, we consider
the case that two codevectors \( y_1 \) and \( y_2 \) in the codebook satisfy the inequality equation in
the following respectively

\[
|d(x, c) - d(y_1, c)| \leq d(x, y_1)
\]

(4.21)

and
\[ |d(x,c) - d(y_2,c)| \leq d(x,y_2) \quad (4.22) \]

\[ |d(x,c) - d(y_2,c)| \geq d(x,y_1) \quad (4.23) \]

then

\[ |d(x,y_2)| \geq d(x,y_1) \quad (4.24) \]

The above description indicates that if the two codevectors \( y_1 \) and \( y_2 \) and satisfy (5), then the distance between \( y_1 \) and input vector \( x \) is greater than the distance between \( y_2 \) and input vector \( x \), and thus cannot become the best match. In other words, we can reject the impossible codevectors using the triangle inequality. This reduces the distance calculation to some simple comparison operations. The fast codebook search using triangle inequality is described in more detail as follows.

For an input vector \( x \), we choose an initial best-match codevector \( y_{bm} \) and the associated distance

\[ d_{\text{min}} = d(x,y_{bm}) \quad (4.25) \]

The codevector, of course, satisfies the following inequality:

\[ |d(x,c) - d(y_{bm},c)| \leq d(x,y_{bm}) = d_{\text{min}} \quad (4.26) \]

codevector in the codebook satisfies

\[ |d(x,c) - d(y_j,c)| \geq d_{\text{min}} \quad (4.27) \]

\[ d(x,y_j) \geq d_{\text{min}} \quad (4.28) \]

where the codevector \( y_j \) can be rejected.

If

\[ \{ y_j : |d(x,c) - d(y_j,c)| \leq d_{\text{min}} \} \quad (4.29) \]
replace \( y_{mn} \) with \( y_i \) and \( d_{mn} \) with \( d(x, y_i) \). Consequently, the search space is further reduced. By doing this recursively, the search space becomes smaller and smaller, thus the encoding complexity can be reduced significantly. With single control point the search space is reduced as shown in figure.

4.6.6 Experiment 3 – fast search VQ based on TIE

VQ based on triangle in equality Codebook size 256

<table>
<thead>
<tr>
<th>Image</th>
<th>Full search</th>
<th>TIE –VQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>32.60</td>
<td>32.60</td>
</tr>
<tr>
<td>Barb</td>
<td>27.89</td>
<td>27.88</td>
</tr>
<tr>
<td>Boat</td>
<td>29.81</td>
<td>29.80</td>
</tr>
<tr>
<td>Mandrill</td>
<td>25.94</td>
<td>25.92</td>
</tr>
<tr>
<td>Goldhill</td>
<td>32.26</td>
<td>27.60</td>
</tr>
<tr>
<td>Testpat1</td>
<td>21.29</td>
<td>21.6</td>
</tr>
<tr>
<td>Testpat2</td>
<td>16.38</td>
<td>12.20</td>
</tr>
<tr>
<td>Hurricane</td>
<td>34.23</td>
<td>34.20</td>
</tr>
</tbody>
</table>

Table 4.5- TIE based VQ

Here we propose new classified VQ based on TIE

Classified VQ based on triangle inequality

Classified VQ reduces number of searches using classifying criterion. While TIE search techniques uses mathematical inequality for reduction in search. Both techniques can be combined to get still better search technique.

Let codebook is arranged in the form of clusters. Each cluster becomes subcodebook. \( C_n \) are the cluster centres or subcodebook centre, where \( n=1,2,...N \)

Where \( N= \) number of clusters.

If \( d(C_i-X) < d(C_j-X) \)

It is more likely that best match for \( X \) will lie in cluster region \( i \) than \( j \).

Therefore we can search only codewectors in subcodebook ‘i’. To find best match in subcodebook region we can use triangle inequality.
This procedure gives extremely quick search at the expense of slightly reduced quality.

![Diagram of TIE based VQ search](image)

**Figure 4.13- TIE based VQ search**

![Diagram of new classified TIE based VQ in wavelet domain](image)

**Figure 4.14- New classified TIE based VQ in wavelet domain**

Classified VQ based on triangle inequality
1. Codebook is generated using training set based on LGB.
2. Codevectors are classified into four clusters using LGB algorithm. I.e. four vectors from codebook are selected as seed vectors. The codebook is partitioned using these seed vectors. Each Codevector is classified to the closest seed vector class. Centroid of the cluster or class becomes new seed
vector. After few iterations no significant change in seed vector is found. Then each cluster is treated as subcodebook.

3. Centroid of the subcodebook are treated as ‘test vectors’.

4. Subcodebook centroid are also considered as control vectors. Distance between centroid (control vector) and codevectors are stored. Number of control vectors is equal to the number of centroids. But distance between each control vector and every codevector need not to be calculated as in case of multiple TIE. Only distance between centroid of the subcodebook and vectors in that subcodebook are stored. This reduces storage space for distance information.

5. To find best match for the input vector, first closest cluster or subcodebook is selected, using minimum distance criterion between test vectors and input vectors.

6. For the selected subcodebook, best match is found using TIE criterion. Because centroid is selected as control vector, many vectors get eliminated and quick search is obtained.

### 4.6.7 Experiment 4 – New Classified VQ based on TIE

Main codebook size 256

<table>
<thead>
<tr>
<th>Image</th>
<th>4 subcodebooks</th>
<th>8 subcodebooks</th>
<th>Full search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>33.27</td>
<td>33.05</td>
<td>34.04</td>
</tr>
<tr>
<td>Barb</td>
<td>28.05</td>
<td>27.73</td>
<td>29.04</td>
</tr>
<tr>
<td>Boat</td>
<td>30.02</td>
<td>30.83</td>
<td>31.25</td>
</tr>
<tr>
<td>Mandrill</td>
<td>25.64</td>
<td>24.49</td>
<td>26.70</td>
</tr>
<tr>
<td>Goldhill</td>
<td>32.53</td>
<td>32.01</td>
<td>32.87</td>
</tr>
<tr>
<td>Testpat1</td>
<td>21.38</td>
<td>21.19</td>
<td>22.87</td>
</tr>
<tr>
<td>Testpat2</td>
<td>16.59</td>
<td>16.34</td>
<td>16.96</td>
</tr>
<tr>
<td>Hurrican1</td>
<td>34.56</td>
<td>34.32</td>
<td>35.45</td>
</tr>
</tbody>
</table>

Table 4.6 – Classified TIE based VQ
4.6.8 Conclusions of experiment 3 and 4

1. Fast VQ, in which search is carried out on TIE principle instead of Euclidean distance is implemented. In full search method, for codebook size of 256, we have to carry 256 searches for every input vector. We have to carry 256 distance calculations for best match.

2. Triangle inequality based search method gives fast codebook search. For codebook of size 256, for every input vector it was found that average codevectors searched are 73. i.e. 183 codevector get eliminated using TIE.

3. By inequality principle we eliminate large number of codevectors and do distance calculations only for 73 vectors on an average. Quality of VQ is maintained same.

4. New classified VQ based on triangle inequality is implemented here. This VQ does not require side information to indicate class codebook being used. Purpose of classification is to speed up the encoding process. Classification criterion is distance between input vector and subcodebook centroid. Classification along with TIE criterion reduces number of searches. Performance of classified VQ is better in terms of searches than VQ, based on only triangle inequality in terms of number of searches. For Lena image for same codebook of size 256 using classified VQ with four codebooks, we need only average 12 searches. However quality is affected. Degradation in quality indicates that method does not give best match like full search. The reason may be that some input vectors get wrongly classified to the improper subcodebook. If centroid of particular subcodebook is at minimum distance from input vector we search only that particular subcodebook for best match. However vector from other subcodebook may be providing better match. Results show that quality degradation is not high, which indicates that most of the vectors are getting correctly classified.

4.6.9 Interpolative VQ

In interpolative VQ method uses different codebooks at encoder and decoder. Smaller feature vector derived from actual vector is used for codebook matching at encoder.
From this index, higher dimensional input vector can be obtained using decoder codebook. Thus there exist direct mapping relationship between encoder codevector and decoder code vector at given index. There are different methods to derive feature vector from input vector.

There exist strong correlation between lower resolution wavelet coefficients to higher resolution level coefficients. This property is very useful for interpolative VQ. 

**Literature survey shows that Interpolative VQ is not well tried by researchers.**

In the paper by Ferki [46] he has discussed the problem of reconstruction of a high resolution image from a lower resolution image by a jointly optimum interpolative vector quantization method. The interpolative vector quantizer maps quantized low dimensional 2x2 image blocks to higher dimensional 4x4 blocks by a table lookup method.

For wavelet crossband VQ low frequency coefficients can be used as feature vector. For nonlinear interpolative higher frequency coefficients can be predicted from this at decoder, which will be stored in decoder codebook.

**We suggest new method for nonlinear interpolative VQ.**

First at encoder cross band vector is formed after 3 level wavelet decomposition, considering same level all orientation coefficients vectors as in expt1. Encoder codebook is derived using following algorithm

**Algorithm for interpolative codebook generation**

1. Form groups of coefficients as ‘same level all orientations’. Thus 3 coefficients at level 3 (one from each orientation), 12 coefficients at level 2 (4 from each orientation) and 48 coefficients at level 1 (16 from each orientation) are grouped.

2. Form three sets of vectors of dimensions 3, 12 and 48 respectively using above groups.

3. Using codebook generated in experiment 2, partition 3 dimensional training vectors in appropriate regions

4. For given codevector find all those training vectors for which this happen to be closest match. In other words they belong to the same partition cell of book1. Find 12 dimensional corresponding counterparts of all these training vectors. Find centroid of them.
5. Similarly find centroid of all those corresponding 48 dimensional training vectors at next level.

6. Combine closest matching codevector from book1 and centroid at level 2 and 1 to form 63 dimensional vectors. Place this 63 dimensional vector at same location at codevector under consideration, in decoder codebook. Repeat procedure for all codevector in book1.

Suppose first code vector from book1 (3 dimensional) is matching with 3 dimensional training vectors say 39, 45, 48, 127 and 268 respectively. Find centroid of 12 dimensional cell formed by 39, 45, 48, 127 and 268th 12 dimensional training vectors, and also 48 dimensional vectors of same index. Form a new 63 dimensional vector such that 3 coefficients from book1, 12 from centroid1 and 48 from centroid 2, write it in new codebook at index of lowest level first code vector matching index. I.e. first location. Combined codebook becomes 63 dimensional. This codebook is now available at decoder. While book1 is used at encoder.

Here whole problem can be considered as if input vector is of dimension 63. out of which only three coefficients are used as feature vector. While original 63 dimensional vector can be derived from decoder codebook. This method increases compression ratio as well as speed of search to great extent.
4.6.10 Experiment 5- New interpolative VQ

Interpolative VQ

1. Interpolative codebook of dimension 63 is derived from 5 training images
2. Input images is decomposed using biorth6.8 filter and vectors are formed by taking coefficients from all levels all orientation.
3. Coarsest coefficients are taken as feature vector
4. Best match is found for the feature vector. From interpolative codebook actual codevector is found. Input codevector is approximated by new codevector.

<table>
<thead>
<tr>
<th>Image</th>
<th>B/p</th>
<th>PSNR</th>
<th>b/pixel</th>
<th>PSNR</th>
<th>b/p</th>
<th>PSNR</th>
<th>b/p</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>.22</td>
<td>32.40</td>
<td>.24</td>
<td>32.77</td>
<td>.28</td>
<td>33.73</td>
<td>.31</td>
<td>34.21</td>
</tr>
<tr>
<td>Boats</td>
<td>.22</td>
<td>30.38</td>
<td>.24</td>
<td>30.55</td>
<td>.28</td>
<td>30.62</td>
<td>.31</td>
<td>30.41</td>
</tr>
<tr>
<td>Barb</td>
<td>.22</td>
<td>28.91</td>
<td>.24</td>
<td>28.78</td>
<td>.28</td>
<td>28.82</td>
<td>.31</td>
<td>29.19</td>
</tr>
<tr>
<td>Mandrill</td>
<td>.23</td>
<td>26.32</td>
<td>.24</td>
<td>26.42</td>
<td>.28</td>
<td>26.45</td>
<td>.33</td>
<td>26.65</td>
</tr>
<tr>
<td>Goldhill</td>
<td>.24</td>
<td>30.88</td>
<td>.24</td>
<td>30.68</td>
<td>.28</td>
<td>31.16</td>
<td>.31</td>
<td>31.44</td>
</tr>
<tr>
<td>Testpat2</td>
<td>.23</td>
<td>16.28</td>
<td>.24</td>
<td>16.74</td>
<td>.26</td>
<td>17.19</td>
<td>.27</td>
<td>17.34</td>
</tr>
<tr>
<td>Testpat1</td>
<td>.22</td>
<td>21.51</td>
<td>.24</td>
<td>21.44</td>
<td>.26</td>
<td>21.97</td>
<td>.27</td>
<td>23.37</td>
</tr>
<tr>
<td>Hurrican1</td>
<td>0.21</td>
<td>32.16</td>
<td>0.20</td>
<td>32.56</td>
<td>0.25</td>
<td>32.48</td>
<td>0.28</td>
<td>33.93</td>
</tr>
</tbody>
</table>

**Table 4.7 - Interpolative VQ**

![Graph comparing interpolative VQ and Crossband VQ](image)

**Figure 4.16 - Comparison of interpolative VQ and Crossband VQ**

### 4.6.11 Conclusions

Novel nonlinear interpolative VQ is implemented. In this scheme small feature vector is derived from large input vector. Smaller feature vector is used to find best match, hence speed of encoding increases. Input vector dimension selected in experimentation is 63. From input vector, feature vector of size 3 is derived. This gives speed up factor more than 20, or in other words encoding time is only 5%. Graph of compression ratio vs PSNR for full search VQ and interpolative scheme is given in fig.3.14. It shows that interpolative VQ gives better quality than full search. Thus speed up by factor of 20 (5% encoding time) is obtained with no loss in quality.
4.6.12 Priority based vector quantization discussion

In case limited bit budget available. To achieve required compression ratio codebook size is to be reduced. But this will also decrease quality.

We can also achieve higher compression ratio by keeping same codebook size by encoding only few important vectors. Thus we can use the available bits to the important data. In case of image compression, important data is the data, which gives less reconstruction error or better quality.

For vector quantization technique if we want to encode vectors as per their importance i.e. giving priority to some vectors, we should be able to identify the important vectors. We can encode only those important vectors to meet bit budget.

In case of wavelet domain vector quantization, vectors are formed using cross band technique. It can be ‘same orientation all levels’ or ‘same level all orientation.’ The vector which posses more energy is more important than vector which posses less energy. If we do not encode low energy vector whatever loss in reconstruction image quality will be less than loss due to high energy vector. Energy of vector is proportional to standard deviation vector. Using this principal we can do priority vector quantization. As follows

1. Decide suitable threshold such as average standard deviation of set of input image vectors.

2. Find standard deviation of each input vector. If it is less than threshold, do not encode the vector. If standard deviation is more, encode the vector.

3. Side information is necessary to tell whether input vector is coded or not.

4. If bit budget is available lower the threshold

This method also increases speed of encoding
4.6.13 Experiment 6- Priority based VQ

codebook size=1024

<table>
<thead>
<tr>
<th>Lena</th>
<th>Threshold 1.5*mean std deviation</th>
<th>Threshold mean std deviation</th>
<th>Threshold 0.6*mean std deviation</th>
<th>Full search entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>Bits/pixel</td>
<td>PSNR</td>
<td>B/pixel</td>
</tr>
<tr>
<td>Lena</td>
<td>34.78</td>
<td>.3</td>
<td>35.39</td>
<td>.34</td>
</tr>
<tr>
<td>Barb</td>
<td>29.23</td>
<td>.31</td>
<td>29.50</td>
<td>.36</td>
</tr>
<tr>
<td>Boat</td>
<td>31.96</td>
<td>.3</td>
<td>32.46</td>
<td>.36</td>
</tr>
<tr>
<td>Goldhill</td>
<td>32.44</td>
<td>.3</td>
<td>33.18</td>
<td>.37</td>
</tr>
<tr>
<td>Mandrill</td>
<td>26.09</td>
<td>.3</td>
<td>26.67</td>
<td>.38</td>
</tr>
<tr>
<td>Testpat1</td>
<td>25.08</td>
<td>.21</td>
<td>25.86</td>
<td>.21</td>
</tr>
<tr>
<td>Testpat2</td>
<td>16.31</td>
<td>.21</td>
<td>16.54</td>
<td>.21</td>
</tr>
<tr>
<td>Hurrican1</td>
<td>36.83</td>
<td>.23</td>
<td>37.41</td>
<td>.26</td>
</tr>
</tbody>
</table>

Table 4.8- Priority based VQ

4.6.14 Conclusion

1. Priority based vector quantization identifies important vectors and gives chance to code them in order of importance. For wavelet domain important vectors are the vectors, which carries more energy. Standard deviation of the vector can be used as estimate of energy. Thus vectors having higher standard deviation value are considered as important vectors.

2. Here we have coded only input vectors having standard deviation more than threshold, while remaining input vectors are ignored i.e. approximated to zero value. Threshold values are reduced from 1.5 time standard deviation to zero. When threshold is zero all input vectors are coded. In practice we can keep initial threshold high after coding vector with standard deviation more than threshold, If more bit budget is available we can reduce threshold of vector selection and can code more vectors.
3. Using priority based VQ with codebook of size 1024 when vectors above mean standard deviation are coded, we got PSNR 35.66 and bits/pixel (including side information cost) rate is 0.41. The same entropy we get with codebook of size 512 but PSNR is 34.45. Thus by coding important vectors using bigger codebook, we get better quality than reducing codebook size to achieve lower bits/pixel.

4.7 Summary

In this chapter vector quantization in wavelet domain is implemented. Two methods of cross band vector formation are studied and compared.

Large encoding time is the drawback of VQ. Using TIE and classification criterion new fast search technique is implemented. This results into very fast search. Encoding time is nearly 6% of the full search. But quality of the reconstruction is reduced.

New nonlinear interpolative VQ is also implemented. This VQ reduces search to 5% and also increases compression ratio. When performance of this VQ is compared to full search quality is found better than full search at same compression ratio.

New Progressive VQ technique ‘priority based VQ’ is implemented. This technique encodes vectors in order of importance for reconstruction.