VECTOR QUANTIZATION

Ömer Nezih Gerek [a], A. Enis Çetin [b]

[a] *Anadolu University, Department of Electrical and Electronics Engineering, 26470 Eskişehir, Turkey.*
Tel.: +90 222 321 3550; fax: +90 222 323 9501; e-mail: ongerek@anadolu.edu.tr.

[b] *Bilkent University, Department of Electrical and Electronics Engineering, 06533 Ankara, Turkey.*

1 Introduction

Quantization is an unavoidable step in representing signals in digital form for computer processing. It is impossible to represent signal samples in infinite precision in computers or digital signal processors. Analog to digital converters assign each data sample a numerical value with finite precision. Therefore, the numerical values must be quantized to the numerical precision of the computer. Apart from the inherent quantization during digitizing signals, a typical digital signal is stored in a compressed form which is generated by a transform/prediction stage followed by quantization and finally entropy coding. If the samples coming from the optional transform/prediction stage are quantized separately, the operation is called "scalar quantization". Consequently, if the samples are grouped to form vectors, their quantization is called "vector quantization" (VQ).

Changing the quantization dimension from one (for scalar) to multi (for vectors) has many important implications. First of all, VQ does not necessarily correspond to rounding of data *values* to coarse levels, any more. VQ stage produces *indices* that *represent* the vector formed by grouping samples. The output index, which is an integer, has little or no physical relation with the vector it is representing, which is formed by grouping real or complex valued samples. The word "quantization" in VQ comes from the fact that similar vectors are represented by the same index. Therefore, many distinct vectors on the multi–dimensional space are quantized to a single vector which is represented by the index. Each index corresponds to a previously decided vector. In that aspect, the number of distinct indices defines the number of quantization *levels*. It is reasonable to argue that the quantization index of a data vector should be selected according to the nearest vector in the set of previously decided vectors (which is called the VQ codebook). As an example,

if the considered vector $\mathbf{x}$ is nearest to an element of the codebook, say $\mathbf{v}_i$, then the VQ output is simply i. In the de-quantization stage, the index i is reconstructed as the vector $\mathbf{v}_i$, so one can say that $\mathbf{x}$ is quantized to $\mathbf{v}_i$.

Assigning indices to a number of vectors has implications other than coding [1], too. Since vectors near to $\mathbf{v}_i$ are indexed as i and those near to $\mathbf{v}_j$ are indexed as j, this automatically provides the clustering information around codebook vectors. Clustering of vectors is commonly used in solving classification problems [2]. Classification of data is a major element of pattern recognition. As a result, many VQ algorithms that are developed for signal coding, have analogous counterparts in the pattern classification and recognition literature. ISODATA [3], k–Nearest Neighbor (k-NN) [4], and Self-organizing feature maps (SOM) [5] are popular clustering methods which have algorithms very similar to those for desining VQ codebooks, such as Max–Lloyd [6],[7] and Linde–Buzo–Gray (LBG) [8] algorithms. Another commonly used VQ application is "color reduction" for images. Many acquisition devices produce color images allocating 8 bits to red, green and blue components of a pixel, respectively. This makes a total of 24 bits/pixel. Due to display buffer limitations or storage requirements, it is desirable to reduce the number of bits to assign to each pixel. This is also done by vector quantizing RGB components to fewer number of indices [9],[10].

This chapter is organized as follows. First the basic concepts of a vector quantizer is presented. The issue of distortion and several metrics used in the design and implementation of a VQ are presented here. Second, properties of minimum distortion VQ and necessary equations for optimality are presented. In the third section, the basic iteration that optimizes the codebook with a given set of data is presented and several VQ codebook design techniques are introduced. Finally, some typical VQ applications are presented at the end of the chapter.

2 Structure of a Vector Quantizer

A vector quatizer consists of two modules; an encoder $\mathcal{E}$, and a decoder $\mathcal{D}$ [1]. The encoder is a module that assigns an index number i to an input vector $\mathbf{x}$. For example, the input vector in Figure 1 consists of 16 elements, and the encoder generates an index number, say, 5. In this example, the input data is in the form of a matrix. The input vector corresponding to the data is obtained from the entries of the matrix by an appropriate scanning of the matrix.

In this module, there are a number of distinct vectors (called the code vectors or codewords, $\mathbf{v}_i$) to form a set (called the codebook, $\mathcal{C}$). The encoder module searches the codebook for the *nearest* match to the input vector. If the i^{th}

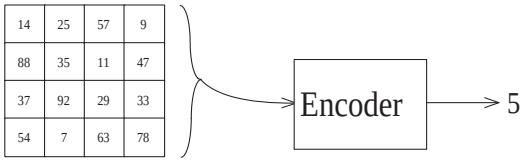


Fig. 1. Encoder produces index=5 to the input vector.

code vector in the codebook ($\mathbf{c}_i$) is nearest to the input vector ($\mathbf{x}$) according to some metric, then the quantizer output is i. The operation is illustrated in Figure 2. As a result of encoding, mere integers are obtained at the output which results in a large amount of representation saving (compression).

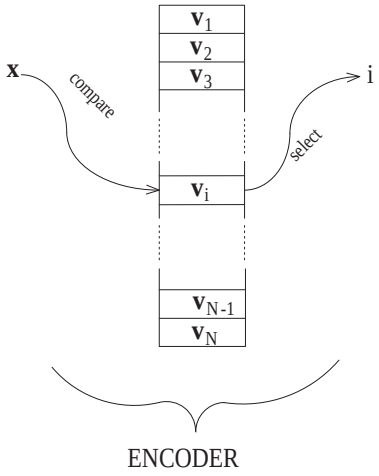


Fig. 2. Comparing the input vector to the code vectors in the codebook.

Since the encoder output is only a representation of code vectors, in order to reconstruct the signal (although with losses), the representation formed by the index numbers must enter to a decoder module which accepts integers at its input and produces the code vector at its output (Figure 3). The decoder is also called the inverse quantizer.

Normally, the term "vector quantizer" (Q) is used for the combination of the encoder and decoder modules. In terms of mathematical notations,

$$\begin{aligned} i &= \mathcal{E}(\mathbf{x}), \\ \mathbf{v}_i &= \mathcal{D}(i), \text{ and} \\ \mathbf{v}_i &= Q(\mathbf{x}) = \mathcal{D}(\mathcal{E}(\mathbf{x})). \end{aligned} \tag{1}$$

The vector quantizer Q has two attributes:

- the dimension, k, and

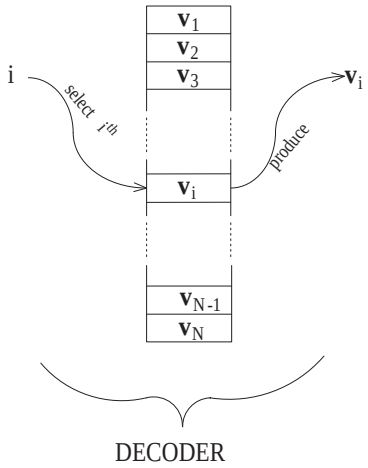


Fig. 3. Generating a code vector according to the input index at the decoder.

- the codebook size, N.

The integer k corresponds to the number of elements in each vector. Therefore, if the elements of the vector are real numbers, an input vector or a code vector is a *point* on the k-dimensional Euclidean space (represented by $\mathbb{R}^k$). The other integer N represents the number of code vectors inside the codebook $\mathcal{C}$. Mathematically, N is called the size of $\mathcal{C}$. In this aspect, Q is an operator from the Euclidean space to a finite set:

$$Q : \mathbb{R}^k \rightarrow \mathcal{C} \tag{2}$$

For coding purposes, the codebook is usually known by both the transmitter (encoder) and receiver (decoder) parts. Therefore, only the integer output (the index) of the encoder is transmitted.

On the other hand, the codebook itself represents a useful partitioning of the k dimensional Euclidean space, $\mathbb{R}^k$ into N regions, R_i. Each region R_i is directly defined by the quantizer in such a way that, if the encoder produces index i for the input vector $\mathbf{x}$, then $\mathbf{x}$ is in region R_i. These regions are also known as *Voronoi cells*. The i-th cluster is, therefore, determined as the set of all vectors in the data set clossest to the the vector $\mathbf{v}_i$. Mathematically;

$$R_i = \{\mathbf{x} \in \mathbb{R}^k | Q(\mathbf{x}) = \mathbf{v}_i\} \tag{3}$$

This means that R_i is the set of all points which are closer to $\mathbf{v}_i$ than to all other code vectors. A region can be bounded (granular cell) with finite k-dimensional volume, or unbounded (overload cell).

Conversely, the above splitting of the k dimensional space into N regions implies an alternative definition of the vector quantizer as follows: If $\mathbf{x} \in R_i$,

then $Q(\mathbf{x}) = \mathbf{v}_i$. Notice that this definition is very suitable when using VQ for grouping or clustering purposes [11]. For these purposes, the encoder indices immediately specify the cluster which the input belongs to.

We have been using the term "nearer to one of the code vectors than others" in the encoder stage, since the beginning of the section. Therefore, what is meant by "nearer" should be clarified. For most practical purposes, an Euclidean distance between two vectors is used for measuring how near two vectors are. On the other hand, we will see that several other distance measures can be used for determining how near two vectors are (Chapter 2 of [12]). The only constraint about the definition of the distance is that, it must be a *proper metric* [1]. If the encoder and decoder uses a proper metric for measuring the distance of the input vector to a code vector in $\mathcal{C}$, then the each region R_i must be convex [2].

Figure 4 shows a 2-D VQ region partitioning. The regions in the central portions are bounded cells, and the ones that extend out of the center (dashed line) are unbounded cells. Notice that this partitioning is proper, in the sense that the regions are all convex.

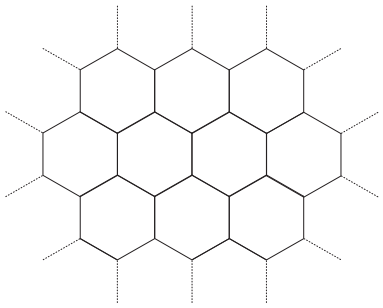


Fig. 4. Typical VQ regions.

The final remark about the general structure of VQ is due to its ability to optimize compression performance for inputs that are grouped to form vectors.

[1] A proper metric is the distance measure $D(\cdot, \cdot)$ for a metric space which satisfies four properties for vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$:

$$\begin{aligned} &\text{nonnegativity: } D(\mathbf{a}, \mathbf{b}) \geq 0 \\ &\text{reflexivity: } D(\mathbf{a}, \mathbf{b}) = 0 \Leftrightarrow \mathbf{a} = \mathbf{b} \\ &\text{symmetry: } D(\mathbf{a}, \mathbf{b}) = D(\mathbf{b}, \mathbf{a}) \\ &\text{triangle inequality: } D(\mathbf{a}, \mathbf{b}) + D(\mathbf{b}, \mathbf{c}) \geq D(\mathbf{a}, \mathbf{c}) \end{aligned} \tag{4}$$

[2] An Euclidean region is convex if the *lines* connecting any two points in the region always lie inside the region, too. A more general definition for convex sets is; if α and β are members of a convex set, then $\lambda\alpha + (1 - \lambda)\beta$ is also a member of the set for $0 \leq \lambda \leq 1$

Shannon has shown that if we have a coding system which maps input vectors into one of N indices with the best coding performance [3], VQ can achieve as good as the above "best" encoder [13]. The way to reach to this performance is through the optimization of the regions and code vectors, which will be described in the next section. Since the minimization is with respect to a distortion, several distortion metrics can be formulated, which all yield different optimization results [14],[15]. Most commonly used metrics are:

- **Minkowski metric:**

$$d_L(\mathbf{x}, \mathbf{v}_i) = \left(\sum_{m=1}^{k} |x(m) - v_i(m)|^L \right)^{1/L} \tag{5}$$

and its special cases:

· **Euclidean (Mean squared error - MSE, $L = 2$) distance:**

$$d_E(\mathbf{x}, \mathbf{v}_i) = \left(\sum_{m=1}^{k} |x(m) - v_i(m)|^2 \right)^{1/2} = (\mathbf{x} - \mathbf{v}_i)^T (\mathbf{x} - \mathbf{v}_i) \tag{6}$$

· **Manhattan (Mean absolute error - MAE, $L = 1$) distance:**

$$d_M(\mathbf{x}, \mathbf{v}_i) = \sum_{m=1}^{k} |x(m) - v_i(m)| \tag{7}$$

· **Chebychev (max, $L = \infty$) distance:**

$$d_C(\mathbf{x}, \mathbf{v}_i) = \max_{m=1}^{k} |x(m) - v_i(m)| \tag{8}$$

- **Hamming distance:**

$$d_H(\mathbf{x}, \mathbf{v}_i) = \sum_{m=1}^{k} \left(1 - \delta_{x(m), v_i(m)}\right), \tag{9}$$

where

$$\delta_{\alpha,\beta} = \begin{cases} 1, \, \alpha = \beta \\ 0, \, \alpha \neq \beta \end{cases}$$

- **Mahalanobis distance:**

$$d_R(\mathbf{x}, \mathbf{v}_i) = (\mathbf{x} - \mathbf{v}_i)^T \mathbf{C}_x^{-1} (\mathbf{x} - \mathbf{v}_i), \tag{10}$$

where $\mathbf{C}_x$ is the autocovariance matrix of $\mathbf{x}$.

[3] The best coder with N output indices is the one that minimizes the distortion between the original vector and the decoder output.

Many other distortion metrics can be developed depending on the application and usefulness. Using a distortion metric, $d(\cdot,\cdot)$, the overall VQ diagram can be re-illustrated as in Figure 5.

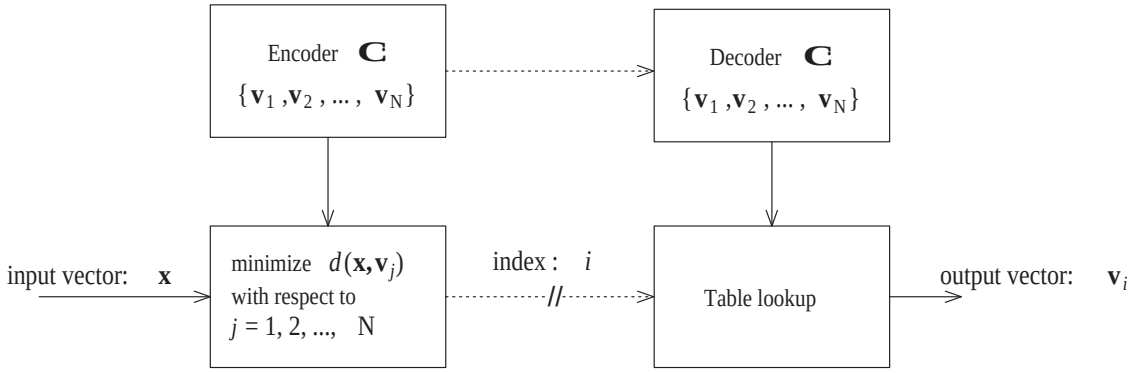


Fig. 5. VQ encoding and decoding process.

3 Minimum Distortion VQ

The performance of a vector quantizer is determined by the optimality of the encoder and decoder parts, described in Section 2. The terms "performance" and "optimum" are directly related to the amount of distortion the quantizer produces at a given number of output levels. In this chapter, we will describe the properties of encoder and decoder parts necessary for a minimum distortion VQ.

3.1 Encoder Optimality

In Section 2, it is pointed out that the index of the output vector is determined according to the minimum distance rule. These kind of vector quantizers are known as "nearest neighbor" quantizers. The "nearest neighbor" rule is required for the optimality of the encoder:

$$Q(\mathbf{x}) = \mathbf{v}_i \text{ only if } d(\mathbf{x}, \mathbf{v}_i) \leq d(\mathbf{x}, \mathbf{v}_j), \forall j \tag{11}$$

The justification of the nearest neighbor rule for encoder optimality is quite simple; if a vector is quantized to a code vector which is not the nearest to the input vector, then the distortion is increased. For that reason, the optimal encoder must search the whole codebook for the smallest distance $d(\mathbf{x}, \mathbf{v}_i)$:

$$d(\mathbf{x}, Q(\mathbf{x})) = \min_{i=1}^{N} d(\mathbf{x}, \mathbf{v}_i) \tag{12}$$

This minimum distortion statistically minimizes the average expected distortion:

$$D = \int d(\mathbf{x}, Q(\mathbf{x})) f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}, \tag{13}$$

where $f_{\mathbf{X}}(\mathbf{x})$ corresponds to the joint *pdf* of $\mathbf{x}$. Consequently, the regions R_i are formed, and the partitioning rule specifies the encoder.

3.2 Decoder Optimality

The second optimality criterion is about the decoder part, satisfied by finding the optimum codebook. In other words, if we are given the clustering regions, we must find the best representing code vector for that region. Statistically, the code vector $\mathbf{v}_i$ in a region R_i must be selected in such a way that the expected distortion it makes with *any* input vector $\mathbf{x}$ that lies inside region R_i must be minimized:

$$\mathbf{v}_i = arg \min_{\mathbf{v}} E \left\{ d(\mathbf{x}, \mathbf{v}) | \mathbf{x} \in R_i \right\} \tag{14}$$

Equation 14 is also known as the "centroid" rule, since the minimization of the expected value corresponds to the centroid of region R_i. For the MSE distortion metric (given in Eq. 6), the centroid corresponds to the minimum MSE (MMSE) estimate of $\mathbf{x} \in R_i$:

$$\mathbf{v}_i = cent(R_i) = E \left\{ \mathbf{x} | \mathbf{x} \in R_i \right\}, \tag{15}$$

where $cent(\cdot)$ stands for the centroid operation.

The proof of Equation 14 is as follows:
The average distortion for a given codebook with a set of code vectors $\mathbf{v}_i$ is

$$D = \sum_{i=1}^{N} \int_{R_i} d(\mathbf{x}, \mathbf{v}_i) f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = \sum_{i=1}^{N} P_i \int d(\mathbf{x}, \mathbf{v}_i) f_{\mathbf{X}|\mathbf{i}}(\mathbf{x}) d\mathbf{x}, \tag{16}$$

where P_i is the probability of $\mathbf{x}$ being in region R_i, and $f_{\mathbf{X}|\mathbf{i}}(\mathbf{x})$ is the conditional *pdf* of $\mathbf{x}$ given $\mathbf{x} \in R_i$. From the definition of the expected value:

$$E \left\{ d(\mathbf{x}, \mathbf{v}_i) | \mathbf{x} \in R_i \right\} = \int d(\mathbf{x}, \mathbf{v}_i) f_{\mathbf{X}|\mathbf{i}}(\mathbf{x}) d\mathbf{x} \tag{17}$$

Since the centroid minimizes the expected value at the left of Equation 17, it also minimizes the integral at the right. Therefore, the summing term of Equation 16, hence the distortion, is minimized.

Equation 15 provides the method to select the code vector corresponding to a region. From this equation, the centroid can be calculated as:

$$\mathbf{v}_i = \int_{R_i} \mathbf{x} f_{\mathbf{X}|\mathbf{i}}(\mathbf{x}) d\mathbf{x} = \frac{\int_{R_i} \mathbf{x} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}}{\int_{R_i} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}}. \tag{18}$$

Finally, an optimum vector quantizer satisfies the following properties:

(i) $E\{Q(\mathbf{x})\} = E\{\mathbf{x}\}$, known as the orthogonality condition,
(ii) $E\left\{\mathbf{x}^T Q(\mathbf{x})\right\} = E\left\{\|Q(\mathbf{x})\|^2\right\}$, and
(iii) $E\left\{\|Q(\mathbf{x})\|^2\right\} = E\left\{\|\mathbf{x}\|^2\right\} - E\left\{\|\mathbf{x} - Q(\mathbf{x})\|^2\right\}$.

4 VQ Codebook Design using Empirical Data

For the optimum VQ, encoder and decoder optimality criteria must be satisfied simultaneously. For a given number of code vectors, say N, the goal is to achieve the minimum distortion by selecting the code vectors, hence the corresponding regions. The properties and equations were described in Section 3. There are a number of methods proposed for achieving the optimal or a sub–optimal quantizer from the given data. One of the most commonly used technique is called the *Generalized Lloyd Algorithm* [6],[7], which improves the codebook iteratively starting from an initial codebook. The scalar version of this quantizer design is also used for scalar quantizer design. Furthermore, this algorithm is commonly referred to as *k-means* [4] or *ISODATA* [3] in the literature concerning clustering.

Lloyd–Max iteration consists of two steps:

- *Nearest Neighbor* condition: Given a set of code vectors, $\mathcal{C} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_N\}$, the clusters are defined as

$$R_i = \{\mathbf{x} \in \mathbb{R}^k | d(\mathbf{x}, \mathbf{v}_i) < (\mathbf{x}, \mathbf{v}_j)\}; \forall i \neq j \tag{19}$$

  If the data $\mathbf{x}$ is on the boundary (with same distortions) of R_i and R_j, assign it to the smaller of i and j.
- *Centroid* condition: Given N clusters, R_i, assign the representative code vector $\mathbf{v}_i$ as the centroid of the cluster. Using the Euclidean distance measure, the centroid corresponds to the arithmetic mean of data vectors belonging

to a cluster. For other distance measures, centroid calculation differs [4].

These two steps are iteratively performed until the overall distortion does not reduce any more, or the amount of distortion improvement goes below a certain threshold after an iteration. Notice that each iteration step must reduce or keep the distortion level. In some cases, empty regions may occur. In that case, a new code vector is assigned, or the codebook size N is reduced.

The Lloyd iteration is a very general method for optimization. However, there are several more VQ design techniques, some of them relying on the Lloyd iteration as intermediate steps. We will name a few of these methods and indicate their basic ideas here.

(a) *Random Coding:* Over a whole set of data vectors, one choses N of the vectors randomly, and assigns them as the code vectors. This is a very empirical technique, however if the data is strongly correlated, it may yield acceptable results.

(b) *Pruning:* In this case, the data vectors are sequentially appended to the codebook list according to whether they are near enough to one of the code vectors in the codebook, or not. If the new vector has a high distance to each code vector, that new vector is added to the codebook [16].

(c) *Pairwise Nearest Neighbor:* The algorithm combines clusters which has nearest centroids, and continues combining as long as the number of clusters is more than the desired codebook size. Initially each data vector forms its own cluster, and the clusters grow iteratively, having more data vectors inside them [17]. The combination of clusters produce a different centroid corresponding to the wighted average of the combined centroids. Therefore, unlike the previous methods, the code vectors do not necessarily correspond to data vectors.

(d) *Product Codes:* If the codebook size is represented as $N = 2^{kR}$, then a cartesian product of k scalar quantizers with 2^R levels can be used as the vector quantizer [18].

[4]

$$
\begin{aligned}
\mathbf{v}_{Euc} &= \frac{1}{M}\sum_{k=1}^{M}\mathbf{x}_k \\
\mathbf{v}_{Man}(i) &= \{\mathbf{x}|P(\mathbf{x}(j) > \mathbf{x}(i)) = P(\mathbf{x}(j) < \mathbf{x}(i))\} \\
\mathbf{v}_{Che}(i) &= \{\min_{j=1}^{M}\mathbf{x}_j(i) + \min_{j=1}^{M}\mathbf{x}_j(i)\}/2 \\
\mathbf{v}_{Ham}(i) &= \{\mathbf{x}_k(i)|P(\mathbf{x}_k(i)) > P(\mathbf{x}_l(i)) \quad \forall l\} \\
\mathbf{v}_{Mah} &= \left[\frac{\sum_{j=1}^{M}\mathbf{C}_{x_j}^{-1}\mathbf{x}_i^T}{\mathbf{C}_{x_j}^{-1}}\right]^T
\end{aligned}
$$

(e) *Lloyd iteration with Stochastic Relaxation:* A zero mean noise is added to the centroids generated by each iteration in the Lloyd algorithm, and the noise power is as the iterations proceed [19]. If the random noise is generated according to a temperature parameter, T_m, which is decreased as iteration number m proceeds, then this technique is also considered as Simulated Annealing.

(f) *Simulated Annealing:* As a subset of the Stochastic Relaxation algorithm, the noise is added to centroids (which is called a perturbation) and the perturbed centroid is accepted with probability $P = e^{-\Delta H/T}$, where ΔH is a cost which increases by the iteration number [20],[21].

(g) *Fuzzy Clustering:* Inclusion of a data vector inside a cluster is not assigned binary values 0 and 1. Instead a fuzzy membership ($S_j(\mathbf{x}_i)$: degree of membership of $\mathbf{x}_i$ in region R_j)value between 0 and 1 is assigned [22]. In this way, the membership of the data vector to the considered region is only partial, and a new fuzzy distortion definition is used: $D_F = \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{N} d(\mathbf{x}_i, \mathbf{v}_j)[S_j(\mathbf{x}_i)]^q$. In the Lloyd iteration, the parameter q is initially selected as a large number (indicating high fuzzyness), and decreased gradually down to 1.

(h) *Linde-Buzo-Gray (splitting) Algorithm:* Probably the most conventional method that utilizes the Lloyd iteration is the Linde-Buzo-Gray (LBG) algorithm [8]. In this case, The algorithm starts with a single code vector (which is normally assigned to be the average of the data vectors. Then the code vector is split into two by adding and subtracting a vector small in magnitude, along the direction of maximum variation in the vector space. With these two new vectors, the Lloyd iteration is applied and optimum code vectors with a codebook size 2 is obtained. LBG algorithm iteratively splits each code vector into two by using the above perturbation method, then applies Lloyd iteration again, until the desired number of code vectors are obtained. This is a very convenient method to completely design the optimal codebook from the scratch without the risk of obtaining empty or unbalanced clusters.

There are other variations on the quantizer design technique, too. For instance, depending on the general structure of the input, it may be desirable to set up a fixed structured quantizer. Lattice vector quantizers are popular for this aspect, where the clusters are selected according to a geometrical grid, mostly hexagonal.

Quantizer improvements are also studied in the literature. The most commonly used improvements can be listed as:

- *Lattice (structured) VQ:* This is actually the "uniform" quantizer in higher dimensions. Each quantization region has the same shape. Therefore, the regions must obey two conditions: (1) They must not overlap, (2) They must cover the N–dimensional input space. Such structures are called *lattice*.

- *Gain-Shape VQ:* If the input data shows significant dynamic range variations, the codebook needs to be very arge for a fairly small distortion. To remedy this situaion, the input vectors can be first normalized and then vector quantized. The normalization factor needs to be encoded separately [1].
- *Mean-Removed VQ:* In many images, the vector segments may contain similar shape charcteristics, but because of intensity variations, they may be quite far from each other according to distance metrics. To quantize such vectors into the same code vector would improve the efficiency. This is possible if the means of the vectors are subtracted from each, and the resulting vectors are quantized. Similar to the above situation, the mean values must be encoded separately.
- *Classified VQ:* If the input data contains multiple patterns that exhibit large spatial differences from each other, while vectors generated from the same pattern portion are quite similar, then designing quantizers separately for each pattern, and applying the appropriate quantizer to the vector improves the efficiency of the quantizer [23]. Usually, there is an overhead of transmitting the infomation of *which* codebook the encoder will use.
- *Multistage VQ:* This method significantly reduces encoder complexity and memory requirements [24]. The idea is to quantize the input coarsely at the first stage, and then continue quantizing the difference between the signal and the its coarsely quantized version, iteratively. As an example, if we have three quantizers Q_1, Q_1, and Q_1, with an input $\mathbf{x}$, then

  $$\begin{aligned}\mathbf{y}_1 &= Q_1(\mathbf{x}) \\ \mathbf{y}_2 &= Q_2(\mathbf{x} - Q_1(\mathbf{x})) = Q_2(\mathbf{x} - \mathbf{y}_1) \\ \mathbf{y}_3 &= Q_3(\mathbf{x} - Q_1(\mathbf{x}) - Q_2(\mathbf{x} - Q_1(\mathbf{x}))) = Q_2(\mathbf{x} - \mathbf{y}_1 - \mathbf{y}_2)\end{aligned}$$

  and the quantization result is $\tilde{\mathbf{x}} = \mathbf{y}_1 + \mathbf{y}_2 + \mathbf{y}_3$.
- *Adaptive VQ:* For purposes such as on–line encoding of signals that change characteristics over time, adaptive VQ is a method to cope with the situation. Usually, the method starts with a relatively large codebook, and selects a subset of the codebook according to the current input characteristics [25].
- *Trellis-Coded Quantization:* Inspired by the trellis coded modulation technique in the communication topic [26], the quantizer uses a quantizer codebook for a given vector which is determined by the VQ output of the previous vector. In that aspect, a data vector can be quantized only after the quantizer output for the previous data vector is determined.

5 VQ Applications and Examples

5.1 Compression

The most widely used application of VQ is data compression [27]. The input data can be compressed by a VQ at the expense of distortion. From an information theoretical perspective, distortion and rate are two inversely proportional quantities. If the compression is high, then distortion increases, but rate decreases. For a given source signal, the rate/distortion curve typically has a shape shown with dashed lines in Figure 6. This curve is obtained by evaluating the minimum distortion achieved by the best encoder at a given rate. On the other hand, the characteristics of a vector quantizer usually looks like the solid line staircase–like shape. Using the optimization techniques described in the previous section, it is desired that the solid lines touch the minimum rate/distortion curve at the given rate.

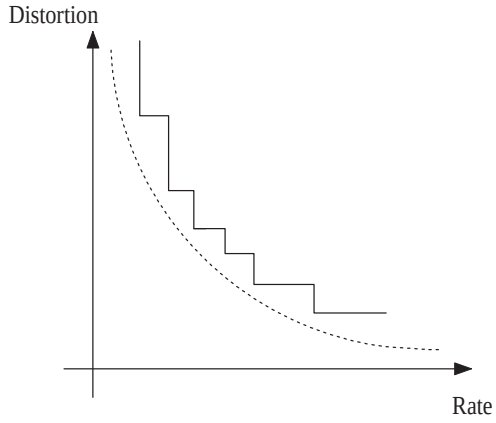


Fig. 6. Rate/distortion curves (dashed: ideal, solid: typical vector quantizer).

5.1.1 Quantization of transform / predictive coding coefficients:

Normally, VQ is the second stage of a conventional compression scheme. The compression is typically composed of

(i) Transformation/Predictive coding stage,
(ii) Quantization, and
(iii) Entropy coding.

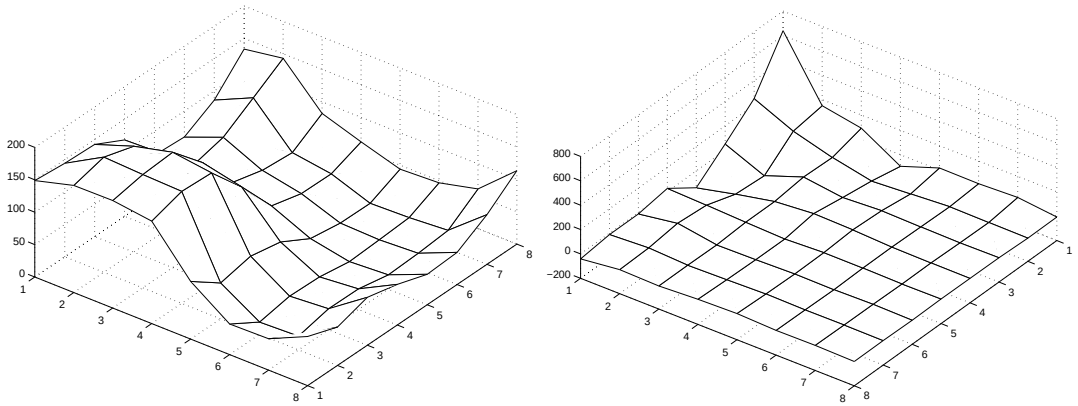
The first stage, transformation or predictive coding, reduces the correlation between samples of the input. Commonly used transforms are the Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT, the celebrated

transform that is used in JPEG), Wavelet transforms, and input specific transforms such as the Karhunen–Loeve Transform and Singular Value Decomposition. The common point in most of these transforms is; they reduce the correlation between the samples of its input vector, hence compacts most of the energy of the vector to only a few of the output vector elements. The operation is reversible by the use of the inverse transforms. Another de–correlating method is called predictive coding, where an element in a sequence is first tried to be predicted using previous elements of the same list, and then only the prediction difference is generated as the output. Using the same prediction algorithm, and given the previously decoded elements, the decoder can re–genrate the same prediction and add the prediction difference to reconstruct the signal.

After either of these methods, the signal samples mostly contain small values, which can be safely quantized to zero. As an example, in the JPEG image compression standard an image is typically divided into 8 by 8 segments and DCT of these segments are computed. This transfom causes a majority of the transform signals to have values that will be quantized to zero. In order to better understand the efficiency of transformation followed by quantization, consider the example of an input vector $\mathbf{x} = [1.2, 1.1, 0.9, 0.8]$. Assume that, in order to achieve some compression, we want to quantize its elements by truncating the samples to the greatest smaller integer ($\lfloor \bullet \rfloor$). If we apply this quantization without any transformation, the output vector would be $\tilde{\mathbf{x}} = [1.0, 1.0, 0.0, 0.0]$, and the distortion would be $\left\{\frac{1}{4}\sum_{i=1}^{4}(\mathbf{x}(i) - \tilde{\mathbf{x}}(i))^2\right\}^{1/2} = 0.6124$. Now, instead of direct quantization, let us first apply DCT to the input signal, and obtain a new vector $\mathbf{c} = DCT\{\mathbf{x}\} = [2.0, 0.3, 0, 0]$. Applying quantization over $\mathbf{c}$, we get $\tilde{\mathbf{c}} = [2, 0, 0, 0]$. Taking the inverse DCT, we obtain $\tilde{\mathbf{x}} = [1.0, 1.0, 1.0, 1.0]$, and the distortion is only 0.1581.

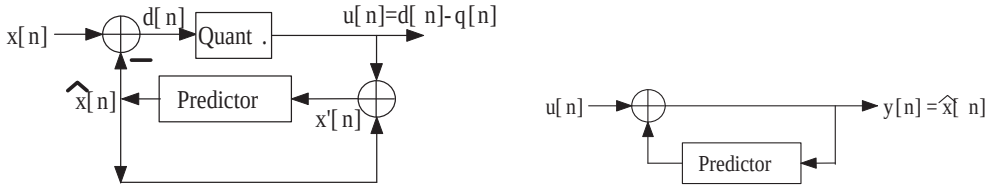
As a second example, consider the 8×8 image shown in mesh format in Figure 7(a). The image values are between 0 and 255 (8 bits). We want to quantize it to 16 levels (4 bits). The quantized version has a distortion of 4.5343, however none of the quantized outputs have a value of the desired zero. The 2D DCT of the same image is shown in Figure 7(b). Notice that most of the coefficients are already very near to zero. If we quantize these coefficients to 4 bits, and then take the inverse 2D DCT, the reconstructed image has a distortion of 3.2112.

Similar to transform coefficients, prediction error samples are also efficiently quantized. However, in this case, the quantization operation should not be applied to the direct output of the prediction step. Instead, the quantizer must be embedded into the prediction module (Figure 8(a))so that the decoder (Figure 8(b))produces the quantized version of the input signal, instead of a diverging signal. The resulting system is called the Differential Pulse Code Modulation (DPCM) [1].



(a) (b)

Fig. 7. (a) An 8×8 segment of an image, and (b) its DCT.



(a) (b)

Fig. 8. DPCM (a) encoder, and (b) decoder.

The recent work on signal compression focuses on efficient vector quantization of subband decomposition / wavelet transformation samples. As examples of state–of–the–art compression algorithms, the commonly used MPEG Audio Layer-3 (known as MP3) standard [28] uses adaptive quantization of subband samples according to the energy of the bands. For images, the Embedded Zerotree Wavelet coder (EZW) [29] and the SPIHT [30] coders use signal dependent grouping and quantization of two dimensional wavelet transform coefficients.

5.1.2 Direct vector quantization of signals

It is also quite customary to apply VQ over sub–blocks of signals, specifically images, without the transformation or prediction. In this case, sub–blocks of an image are taken and fed to one of the VQ design algorithms described in the previous section. One can select the sub–block size $n \times m$ (e.g. 4×4, 8×8, etc.), and the codebook size $N = 2^R$. When the codebook design finishes, each block is represented by R bits. If the original image is $b$bits/pixel (b is usually 8), then the total original $n \times m \times r$ bits will be compressed to R bits.

Consider the 8 bits/pixel 256×256 "Cameraman" image, shown in Figure 9. In order to vector quantize this image, 4×4 and 8×8 block sizes are selected, and codebook sizes of 16 and 32 are tested. LBG and Random Coding algorithms

(described in Section 4) are used as the design methods.

Fig. 9. 256×256, 8 bits/pixel Cameraman image.

Case 1: 4×4 *blocks:* First, let us consider the sixteen 4×4 code vectors generated by the LBG algorithm (shown in Figure 10(a)). It is quite interesting to see that all of the 4×4 sub–blocks of the original image could be represented by one of the vectors in this codebook quite efficiently. Indeed, the total reconstruction distortion is only 17.83 (corresponding to a PSNR of 23.11 dB). The image quantized by this codebook is shown in shown in Figure 11(a). Perhaps, what is more interesting is, the codevectors generated by the Random Coding algorithm (shown in Figure 10(b)) could also produce an acceptable performance of 22.98 dB PSNR (shown in Figure 11(b)). Note that Random Coding has significantly less computational complexity. For both cases, the compression ratio is $CR = (4 \times 4 \times 8) : (log_2 16) = 32 : 1$.

(a) (b)

Fig. 10. Sixteen 4×4 codevectors generated by (a) LBG algorithm, and (b) Random Coding method.

If the codebook size is increased to $N = 32$, this corresponds to using an extra bit. In this way, the distortion would decrease. Figure 12(a) shows 32 code vectors of size 4×4, generated by the LBG algorithm. In Figure 12(b), the quantized image is presented. This image has a PSNR of 24.32.

Case 2: 8×8 *blocks:* Finally, the same compression ratio of 32:1 could also

(a) (b)

Fig. 11. Quantized images with $N = 16$ and block size of 4×4 using (a) LBG algorithm, and (b) Random Coding method.

(a) (b)

Fig. 12. (a) 32 code vectors generated by the LBG algorithm, (b) Quantized image using this codebook.

be reached by using thirtytwo code vectors of size 8×8. The 8×8 code vectors generated by the LBG and Random Coding algorithms are presented in Figures 13(a) and (b) respectively. Using these codebooks, the reconstructed images for LBG and Random Coding quantizers are shown in Figures 14(a) (PSNR=21.83) and (b) (PSNR=21.62) respectively.

(a) (b)

Fig. 13. Thirtytwo 8×8 codevectors generated by (a) LBG algorithm, and (b) Random Coding method.

(a) (b)

Fig. 14. Quantized images with $N = 32$ and block size of 8×8 using (a) LBG algorithm, and (b) Random Coding method.

It can be seen that using a vector block size of 4×4 *(Case 1)* produces better results than using 8×8 *(Case 2)* blocks at the same compression ratio. The reason is, 4×4 blocks exhibit higher inter–pixel correlations than 8×8 blocks do. Therefore, 4×4 code vectors represent the input vectors more efficiently.

5.2 Classification and clustering

Another application of vector quantization is classification. In many pattern recognition applications, automatic clustering of the input data provides the clusters according to which, one decides on the attribute of an input vector [31]. The example descibed in Figures 13(b) and 14(b) can be analyzed from the point of view of classification. Notice that the ground portion inside the image (Figure 14(b)) is quantized to the same code vector (4^{th} row, 3^{rd} column of codebook in Figure 13(b)). Similarly, the sky is quantized to a common code vector (1^{st} row, 1^{st} column of codebook in Figure 13(b)), too. Same is true about the jacket of cameraman, too. Therefore, a portion of the image is classified to a meaningful class according to the VQ output of that portion. In some cases, the VQ outputs may not all be the same, but they may belong to the same set of code vectors. In that case, the classification is again done by checking which set the VQ output belongs to inside the codebook. The situation can be generalized to other classification problems, as well.

Consider the problem of classifying three objects in a picture depicted in Figure 15 as another example. The three types of objects are; large circles, small circles, and ellipses. For this example, the data that can be considered are area and perimeter of the objects. The poblems are; areas of ellipses and small circles are very similar, and perimeters of ellipses and large circles are ver similar. Because of this, one cannot discriminate all three objects using a thresholding (scalar quantization) over either of the data. This example

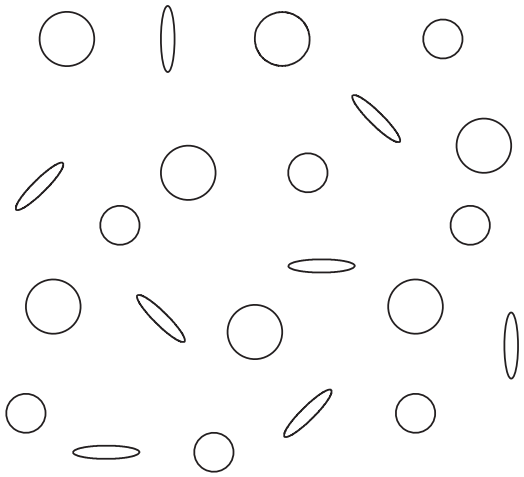


Fig. 15. Three types of objects in an image.

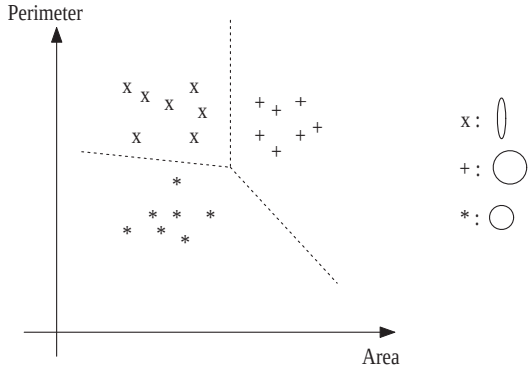


Fig. 16. Area/Perimeter scatter of the objects, and their classification regions.

also indicates that VQ over higher dimensions provide more efficient feature clustering. If both area (a) and perimeter (p) are used together to form the input vector $\mathbf{x} = (a, p)$, then the scatter plot depicted in Figure 16 is obtained. Running a simple 3-level VQ design produces the clustering drawn by the dashed lines. Hence, an efficient classification is obtained.

5.3 Color reduction

A 24–bit true color image can be represented as:

$$x_{m,n} \in \Omega, \tag{20}$$

where $x_{m,n}$ is an image pixel at an arbitrary location (m, n), and Ω is a set defined as: $\Omega = \{(r, g, b) \| 0 \leq r, g, b \leq 255\}$. In this representation, each image pixel is represented by three primary colors: r, g, and b, which all have 8–bits. In a normal color image, the human eye does not distinguish this many colors.

Therefore, there is a representation redundancy. Similarly, many computer monitors (with frame buffers) or printers are not capable of reproducing 24–bit colors. Due to these reasons, it is a desirable application to reduce the number of colors from 2^{24} to, for instance, $2^8 = 256$ [9],[10].

The reduction in the number of colors is normally done by grouping similar colors into a single color index. In that aspect, the operation corresponds to vector quantizing colors (which are vectors with three elements) with a codebook of size 256. One can, therefore, run a VQ optimization program with the input as (r,g,b) pixel values, and output as the indices. Due to computational and speed requirements for this operation, some standard quantizres with fixed clusters are used. This is also called the indexed colormapping, and images with reduced number of colors are called colormap images [32].

A sample color quantizer is illustrated in Figure 17. A group of (r,g,b) vectors with a similar color are grouped to a common color represented by an index, i. A colormap image consists of such indices at the location of each pixel. When a colormap image should be visualized, the display should re–generate (r,g,b) pixel values. As an example, if a pixel $x(m,n)$ has the index i, the decoder must replace the pixel value by the i^{th} code vector inside the codebook. The codebook for this purpose is conventionally called *colormap*. The reconstruction of a pixel color is illustrated in Figure 18.

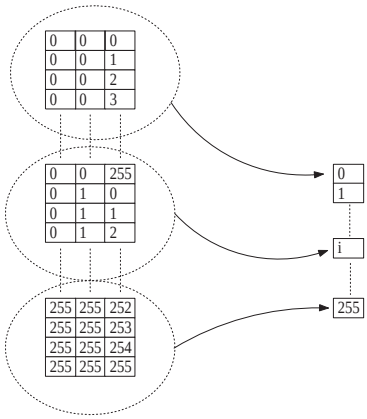


Fig. 17. Quantization of RGB colors into indices.

Color reduction need not be applied on true color images, only. Sometimes, even 256–level gray scale images can be color–reduced to 16 levels. Due to producing fewer number of colors, color reduction must be made carefully according to visual perception parameters. Usually, colormap images tend to exhibit contouring effects around smooth regions (Fig. 19(b)). To produce images with better perceptual characteristics, the customary practice is *dithering*, where a pseudo–random noise is added to the colormap image (Fig. 19(c)).

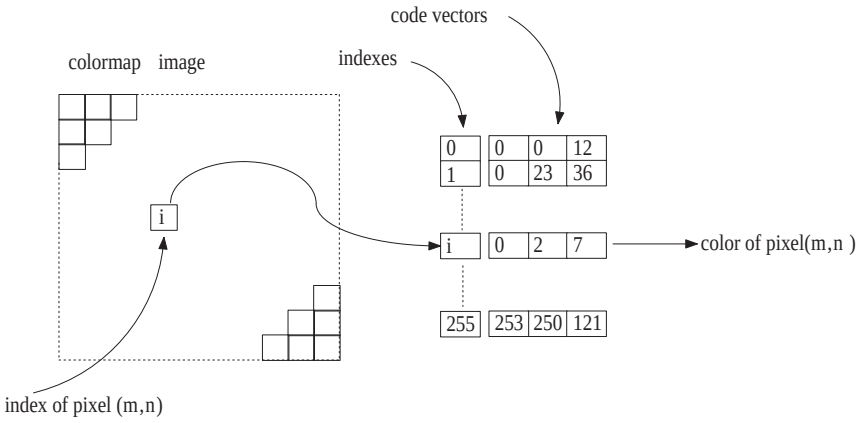


Fig. 18. Reconstruction of RGB color from index.

(a) (b) (c)

Fig. 19. (a) Original 256 level gray scale image, (b) Color reduced to 8 levels by quantization, (c) Color reduced to 8 levels after dithering.

Color reduction is also frequently used in clustering images. A literature survey and application of color reduction in segmenting biomedical images can be found in [33].

References

[1] A. Gersho, R. M. Gray, *Vector Quantization and Signal Compression,* Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[2] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification,* 2^{nd} Ed., John Wiley & Sons, USA, 2001.

[3] G. H. Ball, D. J. Hall, "Isodata, a novel method of data analysis and pattern classification," Stanford Research Institute Report, (NTIS AD699616), Stanford, CA, 1965.

[4] E. A. Patrick, F. P. Fischer, III, "A generalized $k-$nearest neighbor rule," *Information and Control,* Vol. 16, No. 2, pp.128–152, 1970.

[5] T. Kohonen, *Self Organization and Associative Memory,* 3^{rd} Ed., Springer-Verlag, Berlin, 1989.

[6] J. Max, "Quantizing for minimum distortion," *IEEE Trans. on Information Theory,* pp.7-12, March 1960.

[7] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. on Information Theory,* Vol. 28, pp.127-135, March 1982 (unpublished, 1957).

[8] Y. Linde, A. Buzo, R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. on Communications,* Vol. 28, pp.84-95, January 1980.

[9] P. S. Heckbert "Color image quantization for frame-buffer display," ACM *Computer Graphics* (ACM SIGGRAPH '82 Proceedings), Vol. 16, No.3 pp.297-307, 1980.

[10] L. Akarun, Y. Yardımcı, A. E. Çetin, 'Adaptive methods for dithering color images, pp. 950-955, vol.6, no. 7, *IEEE Trans. Image Processing*, July 1997.

[11] A. K. Jain, R. C. Dubes, *Algorithms for clusterin data,* Prentice-Hall Inc., NJ, 1988.

[12] R. M. Gray, *Source Coding Theory,* Kluwer Academic Press, Boston, MA, USA, 1990.

[13] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE National Convention Record, Part 4,* pp.142-163, 1959.

[14] R. D. Short, K. Fukunaga, "Optimal distance measure for nearest neighbor classification," *IEEE Trans. on Information Theory,* Vol. 27, pp.622-627, 1981.

[15] D. L. Davies, D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. and Mach. Int.,* Vol. 1, No. 2, pp.224-227, 1979.

[16] J. T. Tou, R. C. Gonzalez, *Pattern Recognition Principles,* Addison-Wesley, Reading, MA, 1974.

[17] W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. on A.S.S.P.,* pp.1568-1575, October 1989.

[18] M. J. Sabin, R. M. Gray, "Product code vector quantizers for waveform and voice coding," *IEEE Trans. on A.S.S.P.,* Vol. 32, pp.474-488, June 1984.

[19] K. Zeger, A. Gersho, "A stochastic relaxation algorithm for improved vector quantiser design," *Elecronics Letters,* Vol. 25, pp.896-898, July 1989.

[20] A. E. Çetin, V. Weerackody, "Design of Vector Quantizers using Simulated Annealing," *IEEE Trans. Circuits Syst.,* Vol. 35, no. 12, pp. 1550, December 1988.

[21] A. E. Çetin, V. Weerackody, "Design of Vector Quantizers using Simulated Annealing," *The 22th Annual Conf. on Info. Sci. and Sys.,* Princeton University, Princeton, NJ, March 1988.

[22] J. C. Bezdek, "A convergence theorem for the fuzzy ISODATA clustering algorithms," *IEEE Trans. Pattern Anal. and Mach. Int.,* Vol. 3, pp.1-8, 1980.

[23] R. Ramamurthi, A. Gersho, "Classified vector quantization of images," *IEEE Trans. on Communications,* Vol. 34, pp.1105-1115, November 1980.

[24] B. H. Juang, A. H. Gray, "Multiple stage vector quantization for speech coding," *Proc. Intl. Conf. on A.S.S.P.,* IEEE Press, pp.597-600, April 1982.

[25] S. Panchanathan, M. Goldberg, "Adaptive algorithm for image coding using vector quantization," *Signal Processing: Image Communication,* Vol. 4, pp.81–92, 1991.

[26] A. J. Viterbi, J. K. Omura, *Principles of Digital Communications and Coding,* McGraw Hill, New York, 1979.

[27] N. M. Nasrabadi, R. A. King, "Image Coding Using Vector Quantization: A Review," *IEEE Trans. on Communications,* Vol. 36, No. 8, pp. 957–971, Aug. 1988.

[28] ISO/IEC International Standard IS 11172-3 "Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s - Part 3: Audio".

[29] J. M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing,* vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

[30] A. Said and W. A. Pearlman, "An Image Multiresolution Representation for Lossless and Lossy Image Compression," *IEEE Transactions on Image Processing,* vol. 5, pp. 1303–1310, Sept. 1996.

[31] C. Fraley, A. E. Raftery, "How many clusters? Which clustering method? - Answers via model based cluster analysis," *Computer Journal,* Vol. 41, pp. 578–588, 1998.

[32] J. D. Murray, W. vanRyper, *Encyclopedia of Graphics File Formats,* O'Reilly and Associates, Inc., Sebastopol, CA, 1994.

[33] A. E. Raftery, D. C. Stanford, "Determining the Number of Colors or Gray Levels in an Image Using Approximate Bayes Factors: The Psuedolikelihood Information Criterion (PLIC)," Online at: http://www.stat.washington.edu/raftery/.