A SOFTWARE FOR VQ TEACHING

Amarildo Martins de Mattos[1] and Carlos Alberto Ynoguti [2]

Abstract *— This article reports a Matlab® script developed to help students to understand the basic concepts of vector quantization (VQ). It shows in a very intuitive way to design and build a simple two-dimensional vector quantizer. Its graphic interface allows a quick understanding of all the phenomena involved in the quantization process. In addiction, the software makes a comparison between the exhaustive search and a tree-based search to show that the last one has a much better performance.*

Index Terms — Vector quantization, block quantization, source coding.

INTRODUCTION

Quantization and sampling are the basic steps to use the digital technology with analog signals. For unidimensional signals like voice or audio there is the class of scalar quantizers which are exhaustively studied [2] and there are several softwares for their simulation.

Vector quantization can be viewed as a generalization of scalar quantization to the quantization of a vector, an ordered set of real numbers. The jump from one dimension to multiple dimensions is a major step and allows a wealth of new ideas, concepts, techniques and applications to arise that often have no counterpart in the simple case of scalar quantization. While scalar quantization is used primarily for analog-to-digital conversion, VQ is used with sophisticated digital signal processing, where in most cases the input signal already has some form of digital representation and the desired output is a compressed version of the original signal.

VQ is usually, but not exclusively, used for the purpose of data compression. Nevertheless, there are interesting parallels with scalar quantization and many of the structural models and analytical and design techniques used in VQ are natural generalizations of the scalar case.

A vector can be used to describe almost any type of pattern such as a segment of a speech waveform or of an image, simply by forming a vector of samples from the waveform or image. Another example, of importance in speech processing, arises when a set of parameters (forming a vector) is used to represent the spectral envelope of a speech sound. Vector quantization can be viewed as a form of pattern recognition where an input pattern is approximated by one of a predetermined set of standard patterns, or in other words, the input pattern is matched with one of a stored set of templates or codewords.

Vector quantization can also be viewed as a front end to a variety of complicated signal processing tasks, including classification and linear transforming. In such applications, VQ can be viewed as a complexity reducing technique because the reduction in bits can simplify the subsequent computations, sometimes permitting complicated digital signal processing to be replaced by simple table lookups.

Thus VQ is far more than a formal generalization of scalar quantization. In the last years it has become an important technique in speech recognition as well as in speech and image compression, and its importance and application are growing.

In this article, a brief explanation of the basic principles and design methods are provided. In the sequel, a Matlab implementation of a vector quantizer is described.

VECTOR QUANTIZATION

An *N*-level *k*-dimensional quantizer is a mapping *q* that assigns to each input vector $\mathbf{x} = (x_0, \ldots, x_{k-1})$, a reproduction vector $\hat{\mathbf{x}} = q(\mathbf{x})$ drawn from a finite reproduction alphabet, $\hat{A} = \{\mathbf{y}_i; i = 1,...,N\}$. The quantizer is completely described by the reproduction alphabet (or codebook) $\hat{A}$, together with the partition $S = \{S_i, i = 1,...,N\}$, of the input vector space into the sets $S_i = \{\mathbf{x} : q(\mathbf{x}) = \mathbf{y}_i\}$ of input vectors mapping into de i^{th} reproduction vector (or codeword). Such quantizers are also called block quantizers, vector quantizers, and block source codes.

Distortion measures

We assume the distortion caused by reproducing an input vector $\mathbf{x}$ by a reproduction vector $\hat{\mathbf{x}}$ is given by a nonnegative distortion measure $d(\mathbf{x}, \hat{\mathbf{x}})$. Many such distortion measures have been proposed in the literature, and for the sake of simplicity, we'll use the squared error distortion or Euclidean distance in this work. This distortion is given by

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{i=0}^{k-1} |x_i - \hat{x}_i|^2 \qquad (1)$$

[1] Amarildo Martins de Mattos, INATEL National Institute of Telecommunications, Av. João de Camargo, 510, 37540-000, Santa Rita do Sapucaí, MG, Brazil, amartins@inatel.br
[2] Carlos Alberto Ynoguti, INATEL National Institute of Telecommunications, Av. João de Camargo, 510, 37540-000, Santa Rita do Sapucaí, MG, Brazil, ynoguti@inatel.br

Performance

Let $\mathbf{X} = (X_0, \ldots, X_{k-1})$ be a real random vector described by a cumulative distribution function $F_X(x)$. A performance measure of a quantizer q applied to the random vector $\mathbf{X}$ is given by the expected distortion

$$D(q) = E\left[d\left(\mathbf{X}, q(\mathbf{X})\right)\right] \quad (2)$$

where E denotes the expectation with respect to the distribution of $\mathbf{X}$.

Of course, the lower $D(q)$ is, the better is the quantizer. So, an N-level quantizer is said to be optimal (or globally optimal) if it minimizes the expected distortion, that is, q^* is optimal if for all other quantizers having N reproduction vectors, $D(q^*) \le D(q)$. A quantizer is said to be locally optimum if $D(q)$ is only a local minimum, that is, slight changes in q cause an increase in distortion.

The goal of block quantizer design is to obtain an optimal quantizer if possible and, if not so, to obtain a locally optimal and hopefully good quantizer. Several such algorithms have been proposed in the literature, and in this article we will discuss the most popular of them, due to Linde, Buzo and Gray [3], commonly referred as the LBG algorithm. The LBG algorithm is based on the Lloyd's Method I [4], which will described next.

VQ DESIGN

The Loyd's method

The Loyd's Method I is based on the solution of the two following problems:

- Given a quantizer q, described by a reproduction alphabet $\hat{\mathbf{A}} = \{\mathbf{y}_i, i = 1, \ldots, N\}$, what is the optimum partition $S = \{S_i, i = 1, \ldots, N\}$?
- Given a partition $S = \{S_i, i = 1, \ldots, N\}$, where the code words must be placed so that $D(q)$ is minimized?

The solutions for the two problems are:

- A partition that is optimum for $\hat{\mathbf{A}}$ is easily constructed by mapping each input vector $\mathbf{x}$ into the codevector $\mathbf{y_i}$ that minimizes the distortion $d(\mathbf{x},\mathbf{y_i})$, that is, by choosing the minimum distortion or nearest-neighbor codeword for each input.
- For a given partition S, the best location for each codeword is the centroid or center of gravity of each set S_i.

The LBG algorithm

The LBG algorithm builds up the vector quantizer by solving these two problems in a recursive way. The LBG algorithm can be state as follows:

The LBG algorithm

- **Initialization:** Choose an arbitrary set of K code vectors, say $\bar{\mathbf{x}}_k, k = 1, 2, \ldots, K$.
- **Recursion:**
  1. For each feature vector $\mathbf{x}$ in the training set, "quantize" $\mathbf{x}$ into code vector $\bar{\mathbf{x}}_{k^*}$, where

$$k^* = \arg\min_k d\left(\mathbf{x}, \bar{\mathbf{x}}_{k^*}\right) \quad (3)$$

  Here $d(\cdot,\cdot)$ represents some distortion measure in the feature space. In this work, the Euclidean distance, (1) was used.

  2. Compute the total distortion that has occurred as a result of this quantization,

$$D = \sum d\left(\mathbf{x}, q(\mathbf{x})\right) \quad (4)$$

  where the sum is taken over all vectors $\mathbf{x}$ in the training set, and $q(\mathbf{x})$ indicates the codeword to which $\mathbf{x}$ is assigned in the current iteration. If D is sufficiently small, STOP.

  3. For each k, compute the centroid of all vectors $\mathbf{x}$ such that $\bar{\mathbf{x}}_k = q(\mathbf{x})$ during the present iteration. Let this new set of centroids comprise the new codebook, and return to Step 1.

There also exists a slight variation on the LBG method, which differs in the way the algorithm is initialized: the number of clusters is iteratively built up o a desired number (power of two) by "splitting" the existing codewords at each step and using these split codes to seed the next iteration. This modified algorithm is shown below:

Initial guess by splitting

- **Initialization:** find the centroid of the entire population of vectors. Let this be the (only) initial codevector.
- **Recursion:** There are a total of I iterations, where 2^I codevectors are desired. Let the iterations be i = 1, 2, …, I. For iteration i,
  1. "Split" any existing code vector, say $\bar{\mathbf{x}}$, into two code vectors, say $\bar{\mathbf{x}}(1+\varepsilon)$ and $\bar{\mathbf{x}}(1-\varepsilon)$, where ε is a small number, typically 0.01. This results in 2^i new code vectors, say $\bar{\mathbf{x}}_k^i, k = 1, 2, \ldots, 2^i$
  2. For each feature vector $\mathbf{x}$ in the training set, "quantize" $\mathbf{x}$ into code vector $\bar{\mathbf{x}}_{k^*}$, where

$$k^* = \arg\min_k d\left(\mathbf{x}, \bar{\mathbf{x}}_{k^*}\right) \quad (5)$$

  Here $d(\cdot,\cdot)$ represents some distortion measure in the feature space. In this work, the Euclidean distance, (1) was used.

3. For each k, compute the centroid of all vectors $\mathbf{x}$ such that $\overline{\mathbf{x}}_k^i = q(\mathbf{x})$ during the present iteration. Let this new set of centroids comprise the new codebook and, if $i < I$, return to Step 1.

VECTOR QUANTIZATION

Once the vector quantizer is constructed, it's ready for use. There are several ways to perform vector quantization of a input vector, and here two of them will be discussed in details: the exhaustive search and the tree search.

Exhaustive search

The exhaustive search is the most direct and simple encoding algorithm. For this encoding scheme, a codevector is selected by calculating the distortion between the input vector x and all the codevectors in the codebook. The codevector having the minimum distortion is then selected.

Tree search

The exhaustive search outline above requires, for a codebook of size N, N distortion evaluations to be performed. For the squared error distortion measure, it means that k multiplications and (k-1) additions must be performed for each of the N codevectors. Other distortion measures may have much higher computational demands. Frequently the codebook size needed in applications is very large. Furthermore, the vector rate $\boldsymbol{f}_v$ is rather high in typical communications systems and the number of distortion calculations that must be performed per unit time, given by $N\boldsymbol{f}_v$, implies a very demanding computational complexity, typically involving many millions of arithmetic operations per second.

These considerations motivated serious study of more efficient algorithms that yield the nearest codevector without requiring an exhaustive search through the codebook. Several approaches have been proposed, and here, the tree search algorithm will be presented.

Consider the two-dimensional quantizer shown in Figure 1. In this Figure, the six code vectors are indicated with dots and the labeled "hyperplane" decision boundaries (line segments) are indicated with thick lines, which separate neighbor regions. Thin lines indicate extensions of the hyperplane segments to help visualize the efficient coding operation. Each decision boundary is a segment of a hyperplane and is labeled with the letters A, B, C, … and the two half spaces separated by each hyperplane are labeled '+' and '-'. Next, a tree structure for an efficient successive approximation procedure to locate an input vector will be described.

Suppose the initial step of a search algorithm is to compute the binary decision function for the hyperplane A. If the input $\mathbf{x}$ is on the right side of A (labeled '+'), then code vectors 1 and 5 are immediately eliminated as candidates for the nearest neighbor, since they are contained entirely on the left side of A. Similarly, code vectors 2 and 3 are eliminated as candidates if the input is on the left side of A ('-'). Depending on the result of the first test, we choose one of two new tests to perform. Thus if the input lies on the '+' side of A, we then test to see on which side of the hyperplane C it lies. If the input is on the '-' side of A, we then determine of which side of hyperplane B it lies. Each test eliminates one or more candidate codewords from consideration. This kind of procedure corresponds to a tree structure for the search algorithm as shown in Figure 2.

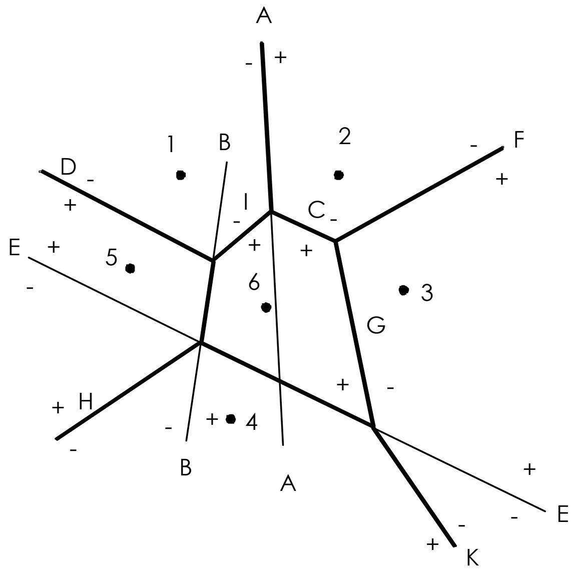

FIGURE 1
A TWO-DIMENSIONAL VECTOR QUANTIZER. (AFTER GERSHO & GRAY [1]).

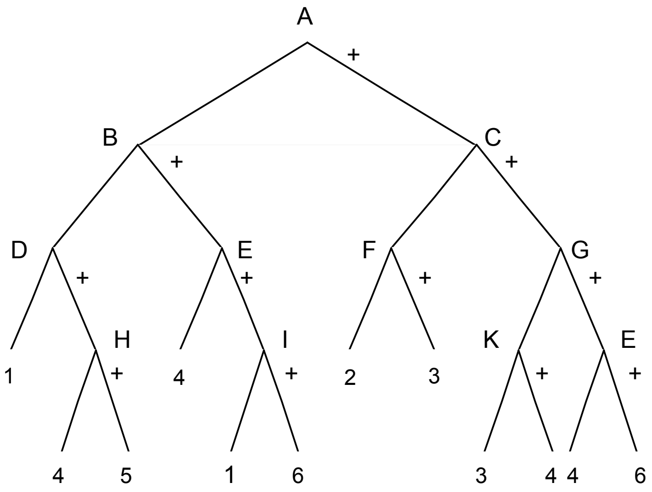

FIGURE 2
TREE STRUCTURE FOR EFFICIENT NEAREST NEIGHBOR SEARCH (AFTER GERSHO & GRAY [1]).

Note that this particular tree always leads to a final determination of the nearest neighbor code vector after at most 4 binary decisions. Hence, the search complexity has been reduced from 6 distance computations to 4 scalar product operations.

MATLAB SCRIPTS

To illustrate these concepts, three Matlab scripts were developed, one for codebooks design, and the others for search time evaluation. The first script implemented the modified LBG algorithm, the second the exhaustive search method, and the third, the tree search procedure. Next, the script outputs are shown.

VQ design

As an illustration of the VQ design using the LBG algorithm, let the training points be as shown in Figure 3.

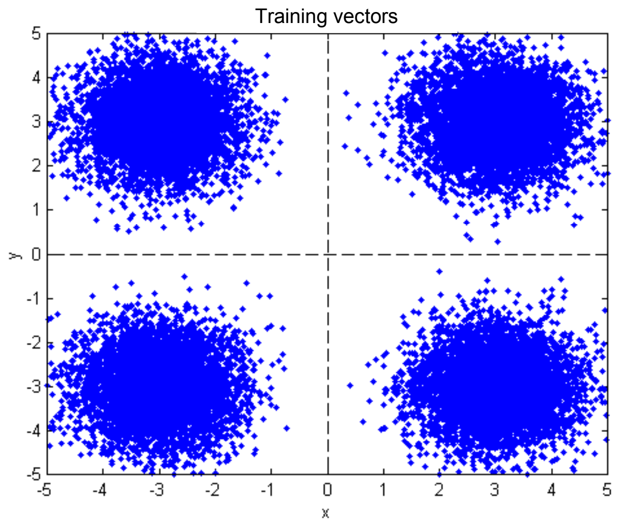

FIGURE 3
TRAINING VECTORS FOR THE VQ DESIGN.

If a codebook with 4 codevectors is desired, it's clear that these codevectors should be located at positions (-3,-3), (3,-3), (-3,3) and (3,3), as shown in Figure 4.

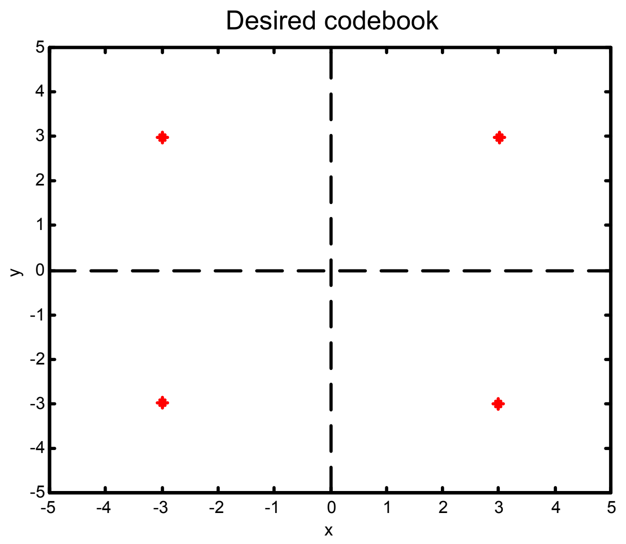

FIGURE 4
DESIRED 4 CODEVECTORS CODEBOOK.

In the first step, only one centroid is to be calculated, as the center of gravity of the entire set of training points. For the points given in Figure 3, these codevector sholud be located somewhere nearby the origin, and the output of the program is shown in Figure 5.

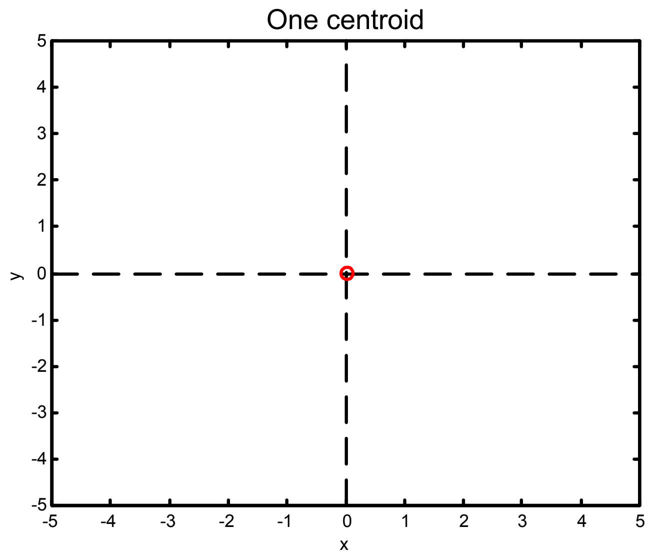

FIGURE 5
A ONE VECTOR CODEBOOK GENERATED BY THE PROGRAM.

Next, this unique codevector should be 'splitted', giving rise to two other ones. The result of the splitting procedure is shown in Figure 6.

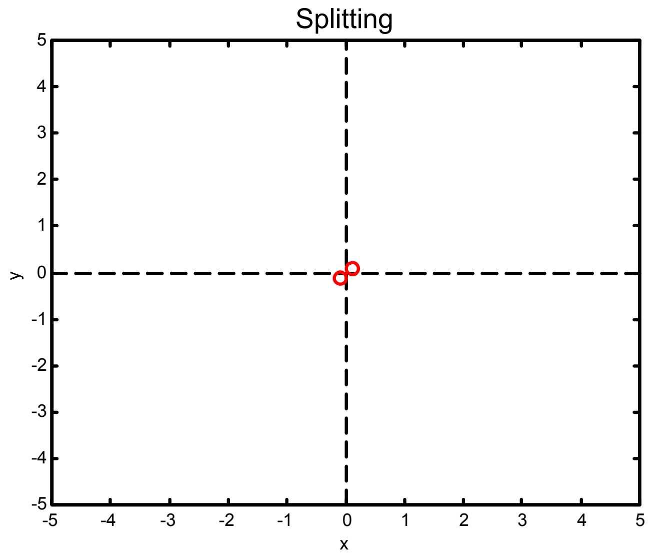

FIGURE 6
CODEBOOK AFTER SPLITTING.

The next step is to reallocate the codevectors so that a minimum distortion codebook is achieved. After a few iterations, the final codebook of order 2 is shown in Figure 7.

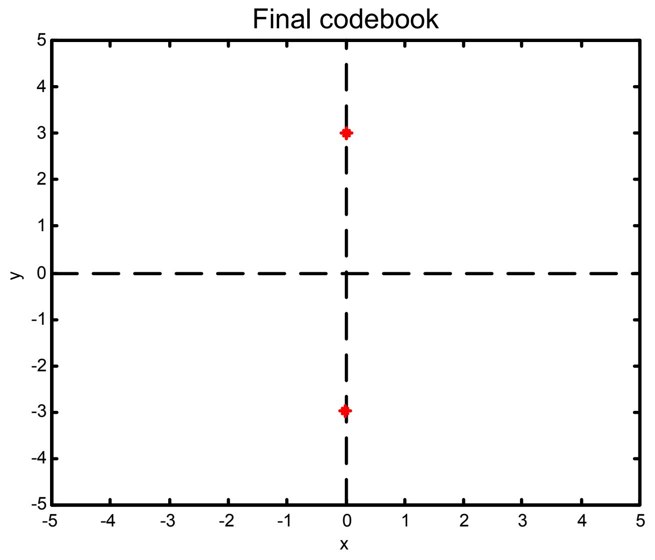

FIGURE 7
FINAL CODEBOOK OF ORDER 2.

Again, the splitting procedure is applied to both the codevectors of this codebook, yielding the configuration of Figure 8

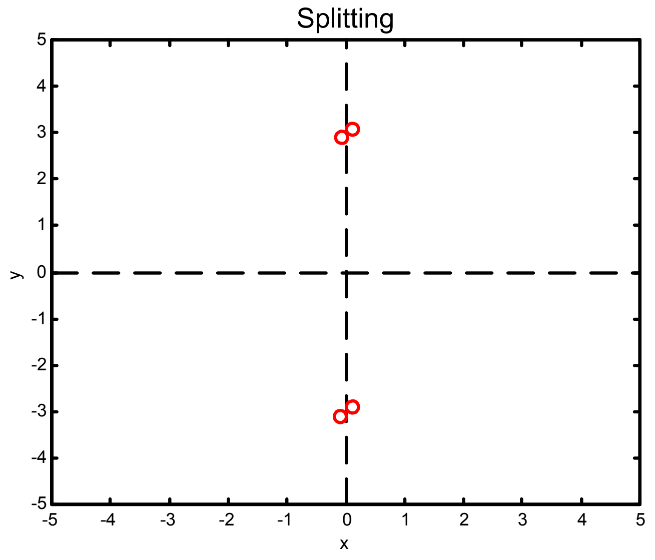

FIGURE 8
CODEBOOK OF ORDER 4 AFTER SPLITTING THE CODEVECTORS OF CODEBOOK OF ORDER 2.

After some iterations, the algorithm leads to the final codebook of Figure 4.

Quantization

After the design of the quantizer, it's time to analyze the quantization performance in terms of quantization time. For this purpose, 10000 points were generated according to a bidimensional uniform distribution, as shown in

The exhaustive search and tree search were then performed, and the quantization times were 1.047 s for the exhaustive search and 0.625 s for the tree search, showing the effectiveness of the later method.

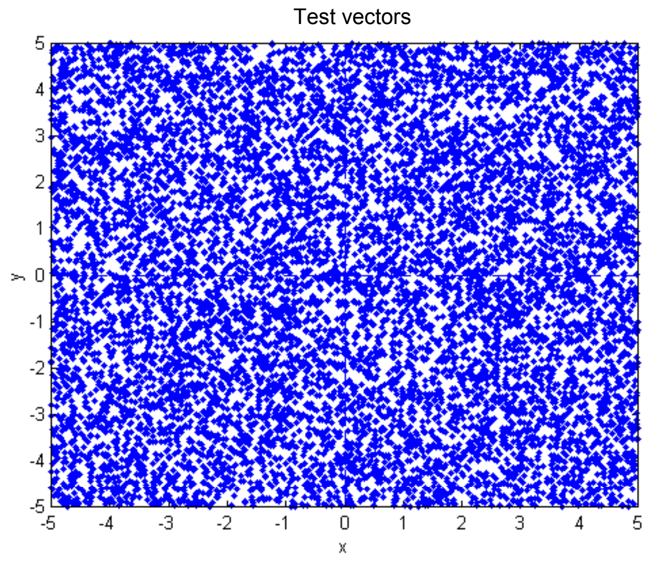

FIGURE 9
POINTS GENERATED FOR SEARCH PROCEDURES TESTS.

CONCLUSIONS

In this article a brief outline of vector quantization theory was presented, together with a Matlab® implementation of the LBG algorithm, the exhaustive search and the tree search, leading to a better comprehension of all phenomena involved.

Computational simulations are a very good way to teach because there are no 'tricks': the students can see the algorithm working and easily understand the concepts involved. As one said: 'A picture say more than a thousand of words'.

For the future, a better user interface is being planned, maybe using the Graphic User Interface provided with Matlab®.

REFERENCES

[1] Gersho, A. and Gray, R. ,"Vector quantization and signal compression", *Kluwer Academic Publishers,* 1991.

[2] Jayant, N.S. and Noll, P., "Digital coding of waveforms – Principles and applications to speech and video", Prentice-Hall, 1984.

[3] Linde, Y., Buzo, A., Gray, R. M., "An algorithm for vector quantizer design". *IEEE Transactions on Communications*. Vol COM-28, No 1. January, 1980, pp. 84-95.

[4] Loyd, S. P., "Least squares quantization in PCM's*", Bell Telephone Laboratories Paper*, 1957.

[5] Deller, J. R., Hansen, J. H. L. and Proakis, J.G., "Discrete time processing of speech signals", IEEE Press, 2000, pp. 70-73.