

# EFFICIENT VECTOR QUANTIZATION USING AN N-PATH BINARY TREE SEARCH ALGORITHM

R. San-Segundo, R. Córdoba, J. Ferreiros, A. Gallardo, J. Colás, J. Pastor, Y. López.

Grupo de Tecnología del Habla. Departamento de Ingeniería Electrónica. Universidad Politécnica de Madrid  
E.T.S.I. Telecomunicación. Ciudad Universitaria s/n, 28040 Madrid, Spain

[lapiz@die.upm.es](mailto:lapiz@die.upm.es)

<http://www-gth.die.upm.es>

## ABSTRACT

We propose the utilization of a new n-path binary tree search algorithm for vector quantization. Our target is to reduce the complexity (time processing) of the vector quantizer maintaining the quantization distortion. The algorithm has been applied to an isolated digit recognizer by telephone based on DHMM and to a continuous speech system based on SCHMM, so we will also give the recognition results for both of them. We have tested several alternatives to calculate the centroids of the higher levels of the tree. In all the experiments we have considered the following parameters for the evaluation: average distortion, same choice percentage, average distortion for the mistakes and processing time. Our reference has been the standard quantization (computing the distance with all centroids). In this reference case the distortion was 220.9 and the processing time was 2.1 seconds. With the n-path binary tree search algorithm, we have obtained a 0.7 seconds processing time with a similar distortion: 226.4. In the semicontinuous system, we have obtained a reduction of 71 % in vector quantization processing time, maintaining the word accuracy.

Keywords: vector quantization, binary tree search, CPU time reduction

## 1. INTRODUCTION

In our systems [1][8], every 25 ms we get 3 vectors with 11 components each: MFCC, first derivative and second derivative parameters vector. We calculate these parameters using Perceptual Linear Predictive filters [3]. Before quantization, the parameter vectors are filtered using RASTA (RelAtive SpecTrAl) [4]. We use independent quantizers for each vector. We will present the results for each quantizer. The experiments were done using 2000 digit pronunciations with about 180 frames per pronunciation including silences. The number of vectors used for each quantizer was about 200.000. These vectors will be our training data to design the codebook.

Assume that  $x = (x_1, x_2, \dots, x_d)$  is a d-dimensional vector whose components  $\{x_k, 1 \leq k \leq d\}$  ( $d = 11$ ) are real-valued, continuous-amplitude random variables. In vector quantization, the vector  $x$  is mapped to another vector  $z$  with discrete-amplitude. It is then said that  $x$  is quantized to  $z$ .  $z$  takes one of a finite set of values  $Z = \{z_i, 1 \leq i \leq d\}$ , where  $z_i = (z_{i1}, z_{i2}, \dots, z_{id})$ . The set  $Z$  is referred to as the *codebook*,  $L$  is the size of the codebook, and  $\{z_i\}$  is the set of codewords or centroids.

## 1.1 Distance

When  $x$  is quantized as  $z$ , a quantization error results and a distortion distance  $d(x, z)$  can be defined between  $x$  and  $z$  to measure the quantization quality. The distortion measure between  $x$  and  $z$  is also known as a distance measure in the speech recognition context. The distance used in our case is the weighted squared Euclidean:

$$d(x, z) = \sum_{i=1}^d w_i (x_i - z_i)^2 \quad (2)$$

where  $x_i$  and  $z_i$  are the components of  $x$  or  $z$ , and  $w_i$  are the unequal weights. These weights are the inverse of the typical deviation of each component.

## 1.2 Training the codebook

To design a codebook, the d-dimensional space of the original random vector  $x$  can be partitioned into  $L$  regions or cells  $\{R_i, 1 \leq k \leq L\}$  and associated with each cell  $R_i$  is a vector  $z_i$ . The quantizer then assigns the centroid  $z_i$  to  $x$ , if  $x$  lies in the region  $R_i$ . This design process is also known as training the codebook. We have used the LBG algorithm [6][5] to train the codebook. As we have 3 quantizers, we need 3 codebooks. In our case, we have considered 3 codebooks with 256 centroids each [2].

## 2. BINARY TREE SEARCH ALGORITHM

In the 'standard quantization' algorithm we compute all the distances between the vector and each centroid of the codebook and choose the nearest one. This process takes a lot of time, so we propose a 'binary tree search' algorithm to obtain the closest centroid to a parameter vector given.

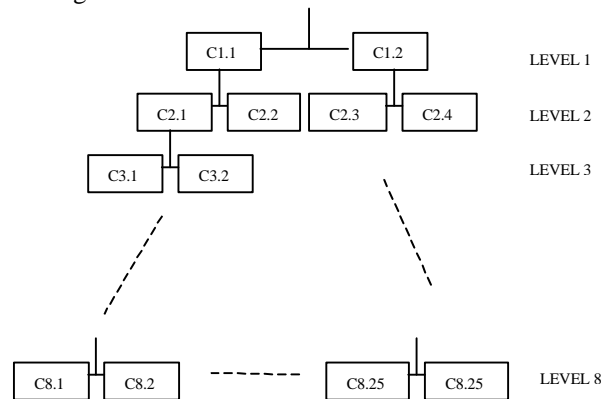


Figure 1: Binary tree of centroids.

Assuming that  $\mathbf{x}$  is the vector to quantize, the binary tree search algorithm begins calculating  $d(\mathbf{x}, C1.1)$  and  $d(\mathbf{x}, C1.2)$ . In level 2, if  $d(\mathbf{x}, C1.1) < d(\mathbf{x}, C1.2)$ ,  $\mathbf{x}$  will be compared with C2.1 and C2.2, otherwise will be compared with C2.3 and C2.4. This process continues repeatedly until level 8 is reached.

With this algorithm, we only need 16 distance calculations. This number is significantly lower than the 256 distance calculations needed in the 'standard quantization' case.

To build the binary tree is necessary to calculate the centroids of the 8 levels. The level 8 centroids (C8.\*) are obtained using the LBG algorithm. The main problem is: How can we get the centroids of the remaining levels?

In this paper, we propose three alternatives and present the results obtained in each one. For each alternative, we are going to measure the following quality parameters:

- Distortion: average distance between the vector to quantize and the closest centroid.
- Same choice percentage: percentage of vectors quantized all right (same result as the 'standard quantization' case)
- Mistakes distortion: distortion in case of wrong quantization.
- Processing time (seconds): average time to quantize a pronunciation ( $\approx 180$  frames). Includes the 3 vectors of parameters.

In Table 1 the results for the 'standard quantization', which we will use as a reference, are shown.

Quality parameters	Code. 1	Code. 2	Code. 3
Distortion	220.9	2591.0	52181.6
Same choice %	100	100	100
Mistakes distortion	0	0	0
Processing time (sec)	2.1		

**Table 1.** Standard quantization

## 2.1 First alternative

Our first option was to use the centroids obtained in each step of the LBG algorithm. The results are shown in Table 2.

Quality parameters	Code. 1	Code. 2	Code. 3
Distortion	421.7 ( $\Delta$ 90%)	5414.9 ( $\Delta$ 108%)	111291.7 ( $\Delta$ 113%)
Same choice %	36.7	24.2	20.6
Mistakes distortion	560.5	6509.8	128921.9
Processing time (sec)	0.073		

**Table 2.** First alternative

The time reduction is significant but the distortion increase is too high. The reason is that in steps 3 and 4, the LBG algorithm can classify and modify several times the codewords, disappearing the dependency between the centroids of different levels.

## 2.2 Second alternative

In this case we have calculated the higher level centroids clustering the level 8 centroids. For example, to compute the level 7 centroids, we cluster in groups of 2 the 256 level 8 centroids.

The clustering criterion is the lowest distance between level 8 centroids (a level 8 centroid can only be clustered once). The level 7 centroid are obtained by averaging the vectors of both clusters and its cluster is the union of the grouped clusters. This process must be repeated until reaching the highest level.

The results are shown in Table 3.

Quality parameters	Code. 1	Code. 2	Code. 3
Distortion	343.9 ( $\Delta$ 51%)	5118.1 ( $\Delta$ 98%)	105826.3 ( $\Delta$ 102%)
Same choice %	49.0	29.0	26.9
Mistakes distortion	534.9	6681.4	136436.4
Processing time (sec)	0.087		

**Table 3.** Second alternative

The distortion decreases but it is far from 'standard quantization'. We still have to consider new alternatives.

## 2.3 Third alternative

This option is like the second one, but the clustering criterion is different: now we look for the lowest cluster distortion *after* the clustering. To calculate the level  $i-1$  centroids we must cluster two level  $i$  centroids. The new cluster will be the union of the two grouped clusters. The new level  $i-1$  centroid is the average vector of the new cluster. A level  $i$  centroid can only be clustered once.

We have considered two options for the distortion:

- *Case A*: consider the distortion as the average distance between the new level  $i-1$  centroid and all the vectors from its new cluster. We obtained worse results, so we will not present them.
- *Case B*: consider the distortion as the average distance multiplied by the size of the cluster. This way, we facilitate the grouping of the smallest clusters, which are the least significant, as we give a greater distortion to the clusters with more vectors.

The results for case B (the best one) are shown in Table 4.

Quality parameters	Code. 1	Code. 2	Code. 3
Distortion	331.1 ( $\Delta$ 50%)	5200.3 ( $\Delta$ 101%)	107395.6 ( $\Delta$ 106%)
Same choice %	47.7	27.3	30.1
Mistakes distortion	490.4	6372.3	139348.2
Processing time (sec)	0.088		

**Table 4.** Third alternative (case B)

These are the best results but they are too far from 'standard quantization'.

The processing time is similar in all options because the quantization algorithm is the same. The distortion obtained depends on the higher level centroids calculation.

You can see in Table 5 the recognition rates in the global task for all the cases so far (for the isolated digit recognizer):

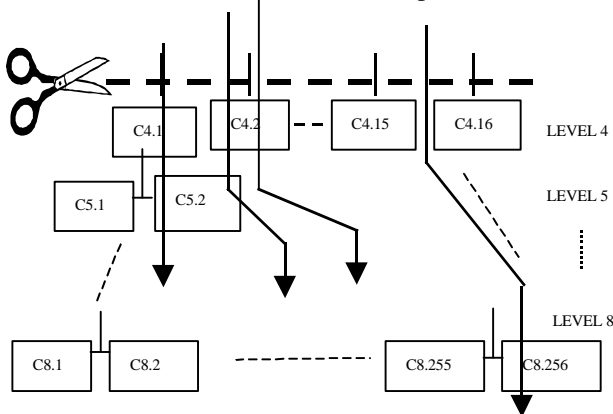
Standard	2.1	2.2	2.3 A	2.3 B
97.1	94.6	95.3	95.3	96.0

**Table 5.** Recognition rates (isolated digit recognizer)

The recognition rate decrement is in all cases very important (considering error rate). The results are not good enough, so we must consider some modifications in the binary tree search algorithm. As we will see, these modifications will reduce the quantization distortion but the processing time will suffer a small increment.

### 3. THE N-PATH ALGORITHM

Maintaining the 3rd alternative, case B, for the calculation of the higher levels centroids, we have considered the following modifications in the binary tree search algorithm in order to decrease the quantization distortion. We will describe them in depth.



**Figure 2:** Binary tree of centroids with modifications.

#### 3.1 First modification

The search algorithm begins in level 4. We find the distance to the 16 centroids of this level. After that, we select the closest one and perform a binary search until level 8 is reached.

This modification increases the processing time because the algorithm needs to compute more distances to reach level 8. The purpose of this modification is to eliminate the decisions on the top levels. These decisions are not very accurate because the centroids of these levels are not very representative. Moreover, a wrong decision in a high level increases the distortion obtained in the level 8.

#### 3.2 Second modification

We decided to maintain several decision paths when the difference between the best distances was low. This means that we can use several paths when the distances are similar and we must select one final path (the centroid chosen) when level 8 is reached. This modification makes less drastic the decisions in higher levels, which are less reliable, increasing the accuracy in the decisions. We have defined 3 new parameters in the algorithm to control the number of paths (or centroids) that we keep in each level of the search:

- MAXPATHS: maximum number of paths considered in each level.
- MINPATHS: minimum number of paths. Needed to assure a good performance.
- PORC: percentage of the minimum distance below which we can consider the path as valid.

The modified binary tree search algorithm can be described as follows (all comparisons are based in distance from the vector to quantize,  $x$ , and one centroid):

1. The first step is to compare the vector  $x$  with all the level 4 centroids (C4.1, ..., C4.16).
2. We calculate and order (by lower distance from  $x$ ) the "MAXPATHS" nearest centroids. Now we decide which ones we are going to keep for the next level.
3. We always keep the "MINPATHS" nearest centroids, as this is the minimum number of paths.
4. If the difference between the distance with the "MINPATHS+1" centroid and the nearest centroid is lower than a PORC percentage of this lowest distance, we keep this path. This process must be repeated with the "MINPATHS+2" centroid, "MINPATHS+3" centroid, ..., until the condition is false or until MAXPATHS paths are reached.
5. Next level: when in level  $i$  there are  $X$  possible paths, in level  $i+1$  we must consider  $2*X$  centroids (their branches in the tree). If we are not in the last level (level 8), we go back to step 2.
6. In the last level one path must be selected for the discrete quantization (DHMM) or  $N$  paths for the soft quantization (SCHMM).

#### 3.3 Experiments

We have made some experiments with these modifications, changing the values of the three parameters. These are the most representative ones:

- A. MAXPATHS: 6, MINPATHS: 2, PORC:  $\infty$ . With these values, a large number of paths are selected. PORC =  $\infty$  implies that the condition to select another path is always true. So, always "MAXPATHS" paths will be considered. The results are shown in Table 6.

Quality parameters	Code. 1	Code. 2	Code. 3
Distortion	226.4 ( $\Delta$ 2%)	2676.6 ( $\Delta$ 4%)	53978.1 ( $\Delta$ 3%)
Same choice %	89.2	88.8	88.2
Mistakes distortion	4914.8	7505.4	103823.1
Processing time (sec)	0.72		

**Table 6.** N-path algorithm. Case A.

B. MAXPATHS: 3, MINPATHS: 1, PORC:  $\infty$ . Reducing the values, less paths are selected, so the processing time decreases. The results are shown in Table 7.

Quality parameters	Code. 1	Code. 2	Code. 3
Distortion	238.8 ( $\Delta$ 8%)	2819.6 ( $\Delta$ 9%)	57132.6 ( $\Delta$ 9%)
Same choice %	81.1	77.1	73.6
Mistakes distortion	2975.9	6345.7	99870.8
Processing time (sec)	0.47		

**Table 7.** N-path algorithm. Case B.

With these modifications we have obtained in both experiments an important reduction in the distortion, resulting in a value similar to the one obtained with the 'standard quantization'.

The processing time is greater than for the pure binary tree search algorithm, but is approximately 3 (case A) and 4 (case B) times lower than for 'standard quantization'.

What is more important is that we can see that the results are quite independent of the parameters: MAXPATHS ranging from 3 to 6 and MINPATHS from 1 to 2 give similar and acceptable results. So, you can tune them in a real-time system to make it quicker if it is needed.

The recognition rates in the global task for the isolated digit recognizer were 97.2 and 97.0 respectively, equivalent to the reference rate (the differences are not significant for the amount of data used).

C. MAXPATHS: 6, MINPATHS: 2, PORC: 50. We have made some experiments changing the value of PORC. The results with PORC = 5000, 500 are similar to the results obtained with PORC =  $\infty$ , even in the processing time, so we will not present them. The results with PORC = 50 are shown in Table 8.

Quality parameters	Code. 1	Code. 2	Code. 3
Distortion	327.6 ( $\Delta$ 50%)	3190.3 ( $\Delta$ 23%)	58737.7 ( $\Delta$ 12%)
Same choice %	67.4	70.0	65.2
Mistakes distortion	1983.0	6374.4	86167.1
Processing time (sec)	0.51		

**Table 8.** N-path algorithm. Case C.

We can see that in this case the distortion is worse and the processing time does not improve much. So we

conclude that this parameter is too sensible for the performance. The recognition result in the global task is 97.2, equivalent to all our previous experiments. This shows that we can accept a worse distortion if we need speed in our real-time system.

For the continuous speech system (SCHMM), in the best case (CASE A), the word accuracy was 74.0 %. This value is into the confidence margin of reference case: 76.5 %  $\pm$  3.1%.

## 4. CONCLUSIONS

The main problem in the binary tree search algorithm is the higher centroids calculation. We have studied some alternatives to calculate these centroids and obtained two approaches with similar results. In any case, the distortion increased in both of them, showing that the reduction was too radical.

The solution has been the n-path binary tree search algorithm, which has divided by four the processing time maintaining a similar distortion, which is an important reduction.

We have found a broad margin where the parameters used can be changed with no significant increase in the distortion.

## 5. REFERENCES

- [1] Córdoba R. "Isolated and continuous recognition systems: comparison and optimisation of the modelling and parametrisation systems". Doctoral Thesis, ETSIT, Madrid, UPM, 1995, pp. 27-58.
- [2] Córdoba R., Menéndez-Pidal X., Macías-Guarasa J., Gallardo A. "Development and improvement of a real-time ASR system for isolated digits recognizer in Spanish over the telephone line". Eurospeech 95, pp. 1537-1540 Madrid.
- [3] Hermansky. H. "Perceptual linear predictive (PLP) analysis for speech" J. Acoust. Soc. Amer., pp. 1738-1752, 1990.
- [4] Hermansky. H., Morgan. N. "Rasta processing of speech". IEEE Trans. on speech and audio processing. V.2, OCT 94.
- [5] Huang. X. D, Ariki. Y., Jack. M.A. "Hidden Markov Models for speech recognition". Edinburg University Press. Edinburg. 1990, pp. 111-119.
- [6] Linde. Y., Buzo. A., Gray. R.M., "Vector quantization in speech coding", IEEE Trans. Comm., v 28, 1980, pp.84-95.
- [7] Ramasubramanian. V. and Paliwal. K. K. "Reducing the complexity of the LPC vector quantizer using the K-D tree search algorithm", Eurospeech 1995, pp. 1255-1259.
- [8] San-Segundo. R. "Optimisation of a telephone-based isolated speech recognition system in a PC" ETSIT, Madrid, UPM, 1997, pp. II.1-24, IV.16-30.