HIERARCHICAL CLASSIFICATION IN VECTOR QUANTIZATION

A.Benazza and G.Tziritas

Laboratoire des Signaux et Systèmes
C.N.R.S. / E.S.E. / U.P.S.
Plateau de Moulon – 91192 Gif–sur–Yvette Cédex

ABSTRACT

This paper proposes to organize the codebook of any vector quantizer, according to a binary tree–structure. Unlike the classical approach, the arborescence is built in an ascending way, using data classification methods. For this purpose, several Sequential Agglomerative Nonoverlapping algorithms (SAHN) are performed. A new encoding process is proposed to improve the VQ performance. It is based on a perceptron algorithm. Simulation results are provided for the selected SAHN methods.

1 INTRODUCTION

In the last decade, vector quantization (VQ) has been found to be a powerful data compression technique which has become very popular in speech and image coding [1]. According to Shannon's rate–distortion theory [2], it is known that VQ achieves better performance than the scalar quantization. For brevity, we denote both vector quantization and vector quantizer by VQ, the context will show if VQ stands for vector quantizer or vector quantization.
A VQ is a mapping Q from $\mathbb{R}^k$ to a given codebook $\mathcal{C} = \{\mathbf{c}_i\}_{i=1,\ldots,N}$ of N k–dimensional codevectors, under some distance measure $d(\cdot,\cdot)$.
In VQ, the first key step is the design of an optimal codebook, minimizing the expected distortion $\mathrm{E}[d(\mathbf{x}, Q(\mathbf{x}))]$. For the traditional squared–error, the centroid and nearest neighbor rules are necessary conditions for optimality [3]. The popular LBG algorithm [4] decreases iteratively the distortion in an attempt to satisfy the latter rules. Such algorithm does not need *a priori* to know the source statistics. Therefore, a long training sequence $\mathcal{S} = \{\mathbf{x}_l\}_{l=1,\ldots,L}$ of L vectors is used. In this context, the LBG algorithm performs a partition $\{\mathcal{S}_i\}_{i=1,\ldots,N}$ of $\mathcal{S}$ according to :

$$\mathcal{S}_i = \{\mathbf{x}_{l_i} \in \mathcal{S} /\ d(\mathbf{x}_{l_i}, \mathbf{c}_i) \leq d(\mathbf{x}_{l_i}, \mathbf{c}_j)\, \forall j \neq i\}; \qquad (1)$$

$$\mathbf{c}_i = \frac{1}{L_i} \sum_{l_i=1}^{L_i} \mathbf{x}_{l_i} \quad \forall\, i = 1, \ldots, N \qquad (2)$$

where L_i is the number of vectors $\mathbf{x}_{l_i}$ in $\mathcal{S}_i$. Unfortunately, the LBG algorithm does not provide any structure for $\mathcal{C}$. Therefore, finding the nearest neighbor $Q(\mathbf{x})$ of a test vector $\mathbf{x}$ implies a full–search procedure which requires N vector distance computations $d(\mathbf{x}, \mathbf{c}_i)$. The codebook size controls the bit–rate r bits/sample since $N = 2^{kr}$. Thus, the complexity of encoding $\mathbf{x}$ increases exponentially with k and r. This exponential growth restricts the applicability of VQ. To circumvent this problem, $\mathcal{C}$ can be constrained to have a binary tree–structure (BTS) as in [5]. Thus, the encoding process is accelerated since the search complexity grows linearly with r. However, the BTS constraint involves a loss of performance. The hope is to reduce the complexity to compensate the degradation. This paper proposes to perform a method of generating a BTS codebook, using the Sequential Agglomerative Nonoverlapping (SAHN) techniques. The first section describes such grouping methods in the context of the VQ. Section 2 deals with the problem of the encoding process. Finally, simulations results are given, in section 3.

2 SAHN METHODS

2.1 Principles

The objective to build a BTS over a given unstructured codebook $\mathcal{C}$, using SAHN approaches. The N triplets $(\mathbf{c}_i,\ L_i,\ \mathcal{S}_i)$ constitute a specific data system. SAHN techniques start with these disjoint clusters $\mathcal{S}_i$. The first step is to select the pair of cells the most "similar" which, when merged, is viewed as a single cluster. The $N-1$ remaining cells are then examined to the next fusion. The procedure is continued untill all the initial N members are in one group. Finally, a sequence of $N-1$ nested clusters is generated. Note that such procedure assumes to get some measure of "closeness" or "similarity" between two groups. This is achieved by defining an inter–clusters distance, denoted $\delta(\cdot,\cdot)$. Several definitions are avalaible according to the desired merging strategy. As for us, we choose the following six SAHN methods [6], [7]:

- Single linkage method

$$\delta_1(\mathcal{S}_i, \mathcal{S}_j) = \min_{\mathbf{x}_{l_i} \in \mathcal{S}_i, \mathbf{x}_{l_j} \in \mathcal{S}_j} d(\mathbf{x}_{l_i}, \mathbf{x}_{l_j}) \quad (3)$$

- Complete linkage method

$$\delta_2(\mathcal{S}_i, \mathcal{S}_j) = \max_{\mathbf{x}_{l_i} \in \mathcal{S}_i, \mathbf{x}_{l_j} \in \mathcal{S}_j} d(\mathbf{x}_{l_i}, \mathbf{x}_{l_j}) \quad (4)$$

- Group average method

$$\delta_3(\mathcal{S}_i, \mathcal{S}_j) = \frac{1}{L_i L_j} \sum_{\mathbf{x}_{l_i} \in \mathcal{S}_i, \mathbf{x}_{l_j} \in \mathcal{S}_j} d(\mathbf{x}_{l_i}, \mathbf{x}_{l_j}) \quad (5)$$

- Ward method

$$\delta_4(\mathcal{S}_i, \mathcal{S}_j) = \frac{L_i L_j}{L_i + L_j} d(\mathbf{c}_i, \mathbf{c}_j) \quad (6)$$

- Variance increase method

$$\delta_5(\mathcal{S}_i, \mathcal{S}_j) = \frac{L_i L_j}{(L_i + L_j)^2} d(\mathbf{c}_i, \mathbf{c}_j) \quad (7)$$

- Nearest centroids method

$$\delta_6(\mathcal{S}_i, \mathcal{S}_j) = d(\mathbf{c}_i, \mathbf{c}_j). \quad (8)$$

One attractive feature of such methods is that they are linear combinatorial. Indeed, a recurrence formula [8] allows to update the distance between the recent merged cluster $\mathcal{S}_i \cup \mathcal{S}_j$ and the remaining ones $\mathcal{S}_k$. Its expression is

$$\begin{aligned}\delta(\mathcal{S}_k, \mathcal{S}_i \cup \mathcal{S}_j) = \\ \alpha_{k,i}\delta(\mathcal{S}_k, \mathcal{S}_i) + \alpha_{k,j}\delta(\mathcal{S}_k, \mathcal{S}_j) + \beta_{i,j}\delta(\mathcal{S}_i, \mathcal{S}_j) + \\ \gamma_{i,j,k}|\delta(\mathcal{S}_k, \mathcal{S}_i) - \delta(\mathcal{S}_k, \mathcal{S}_j)| \quad \forall k \neq i, k \neq j. \quad (9)\end{aligned}$$

Reference [7] gives the parameters for the considered $\delta(\cdot, \cdot)$. Thus, the computational burden is mainly dedicated to the calculation of the $\frac{N(N-1)}{2}$ initial values of $\delta(\cdot, \cdot)$. The next values are then obtained by (9) which reduces dramatically the complexity of these clustering procedures.

2.2 Algorithm Flow Chart

The considered strategies can be now summarized in a general algorithm described below.

- **step 0 (initialization)**
  Begin with the disjoint clustering ($\mathbf{c}_i$, L_i, $\mathcal{S}_i$)
  Compute the pairs $\delta(\mathcal{S}_i, \mathcal{S}_j) \quad \forall i < j$.

- **step 1**
  Select the most similar pair in the current clustering, say $(\mathcal{S}_i, \mathcal{S}_j)$.
  In case of ties, select the first pair encountered.
  If necessary, calculate the centroid of the new cluster $\mathcal{S}_i \cup \mathcal{S}_j$.

- **step 2**
  Update the distances between the two clusters $\delta(\mathcal{S}_k, \mathcal{S}_i \cup \mathcal{S}_j), \forall k \neq i, k \neq j$, using the recurrence formula, given in eq.(9).

- **step 3**
  If only one cluster remains, stop. Else go to step 1.

3 ENCODING PROCESS

3.1 Discriminant Function

Once the $N-1$ fusions performed, the VQ must cover the BTS to encode a test vector $\mathbf{x}$. More precisely, at each node, a rule is requested to select one of the two possible candidate cells $\mathcal{S}_i$ and $\mathcal{S}_j$. This is a discriminant problem and therefore, a discriminant function has to be determined. For sake of simplicity, only linear functions defined as

$$D(\mathbf{x}) = \mathbf{a}_{i,j}{}^{\mathrm{T}}\mathbf{x} + b_{i,j} \quad (10)$$

are considered. The related decision rule is

$$\begin{aligned}\text{if} \quad D(\mathbf{x}) > 0, \quad \mathbf{x} \in \mathcal{S}_i; \quad (11)\\ \text{if} \quad D(\mathbf{x}) \leq 0, \quad \mathbf{x} \in \mathcal{S}_j.\end{aligned}$$

The problem is to find at each layer, the ($\mathbf{a}_{i,j}$, $\mathrm{b}_{i,j}$). Since any knowledge of the clusters shape is avalaible, a learning process is investigated. In this context, a perceptron based error–correction procedure is performed on the training sequence, to adjust the position of the discriminant hyperplane. More precisely, a gradient–descent algorithm (the fixed–increment algorithm [9]) is used, at each non–terminal node. Note that the traditional nearest matching codevector implies a particular position of the separator hyperplane. As such, the described encoder might outperform the usual tree–structured encoder [5]; this is validated by the simulations. This fact shows the major advantage of the considered linear categorizer. Thus, the encoding procedure is operated, according to the decision rule, given in eq.(11). The index of the last layer is then transmitted.

3.2 Encoding Complexity

The SAHN methods generate unbalanced trees. Therefore, the followed path (in other words, the number of crossed layers) depends on both the considered arborescence and the query vector. The tree rate describes the intrinsic structure of the hierarchy. Its expression is

$$R_{tree} = \sum_{i=1}^{N} \frac{L_i}{L} h(i) \quad (12)$$

where $h(i)$ is the height of the terminal node i, e.g. the minimum number of layers crossed from the root to reach the terminal node i. Besides, the average number of comparisons NBL_{test} required to code a given test sequence measures the complexity of the process. The equality $R_{tree} = NBL_{test}$ holds when the training set encoding is accurate. Thus, for a test sequence, the difference $\mid R_{tree} - NBL_{test} \mid$ is related to both the fusion strategy and the discriminant function adopted.

4 SIMULATIONS RESULTS

Simulation experiments are performed using two typical video–conferencing scenes, "White Trevor" and "Miss America". The images are 360 x 288 by 8 bits (respectively, at 15 frames/s and 10 frames/s). The training sequence issues from their partition into $L = 199440$ nonoverlapping square blocks of size 4 x 4 pixels ($k = 16$). The selected objective performance indicator is the peak signal–to–noise ($PSNR$) defined as

$$PSNR(dB) = 10\log(\frac{255^2}{< d(\mathbf{x}, Q(\mathbf{x})) >}) \qquad (13)$$

where $< \cdot >$ denotes averaging over the entire sequence. Codebooks at different bit–rate are generated by the LBG algorithm, initialized either randomly or by the "splitting" technique. These codebooks are then hierarchized by SAHN algorithms. We give only performances inside the training set. The same results are obtained for the two sequences. Only those related to "Miss America" are detailed.

As expected, the discriminant function chosen outperforms clearly the traditional "nearest neighbor codevector". Fig. 1 confirms this for δ_3. Indeed, at the bit–rate $r = 0.625$ bpp, a gain of 6.5 dB can be achieved. Besides, our simulations lead to the rejection of the linkage methods as illustrated in Fig. 2. On one hand, δ_1 tends to yield rather elongated clusters (e.g. very deep trees), increasing the probability of misclassification. This is the well known chaining effect. On the other hand, the complete linkage favors compact groups but it gives bad performances. These phenomena are depicted on the Fig. 3 where high (resp. low) values of R_{tree} are produced by δ_1 (resp. δ_2). In addition to the low $PSNR$, a computational load is required in evaluating the initial inter–clusters distances. These are the reasons why we tend to exclude the linkage methods. The same conclusion is valid for the increase variance method. δ_5 penalizes the big cells. The misclassification of vectors belonging to big cells is frequent and induces very small $PSNR$. At the opposite, the remaining strategies perform much better (Fig. 4). Note that the classical Tree–Structured VQ, introduced in [5] gives comparable $PSNR$ (Fig. 5).

Finally, the degradation of the performance, respect to the full–search is attributable to both the BTS constraint and the assumption of linear separability of the disjoint clusters, at each node. However, such loss of performance is compensated by a dramatical reduction of the complexity (Fig. 6).

5 CONCLUSION

The Sequential Agglomerative Hierarchical Nonoverlapping technique is depicted in this paper. More precisely, six combinatorial strategies are tested to organize unstructured codebooks according to a binary tree–structure. Such structure constraint has the advantage to reduce the encoding procedure complexity. In practice, there is no major difficulty to implement the SAHN methods. Good performances are achieved especially when a perceptron's error correction works as a learning process. Furthemore, these performances are very close to those of the well known Tree–Structured VQ. The problem of adapting the BTS, in the context of image sequence VQ are the matter of our future investigations.

References

[1] R.M.Gray, "Vector quantization," *IEEE. ASSP. Mag.*, vol. 1, no. 2, pp. 4–29, Apr. 1984.

[2] C.E.Shannon, "A mathematical theory of communications," *Bell Systems Technical J.*, vol. 27, pp. 379–423, pp. 623–656, 1948.

[3] A.Gersho, "On the structure of vector quantizers," *IEEE. Trans. Inform. Theory*, vol. IT–28, no. 2, pp. 157–166, March 1982.

[4] Y.Linde, A.Buzo, R.M.Gray, "An algorithm for vector quantizer design," *IEEE. Trans. Commun.*, vol. COM-28, no. 1, pp. 84–95, Jan. 1980.

[5] A.Buzo, A.H.Gray Jr., R.M.Gray, J.D.Markel, "Speech coding based upon vector quantization," *IEEE. Trans. Acoust., Speech, Signal Processing*, vol. ASSP–28, pp. 562–574, Oct. 1980.

[6] R.M. Cormack, "A review of classification," *J. of The Royal Stat. Soc.*, serie A, vol. 134, pp. 321–367, 1971.

[7] E.Diday, J.Lemaire, J.Pouget, F.Testu, *Eléments d'analyse de données.* Paris: Dunod, 1982, Ch. 2, pp. 73–138.

[8] G.N.Lance, W.T.Williams, "A general theory of classificatory sorting strategies: II. Clustering systems," *Comput. J.*, vol. 10, pp.271–277, 1967.

[9] T.Y. Young, T.Calvert, *Classification estimation and pattern recognition*, Elsevier Publishing Co., Inc., 1974, Ch. 4.

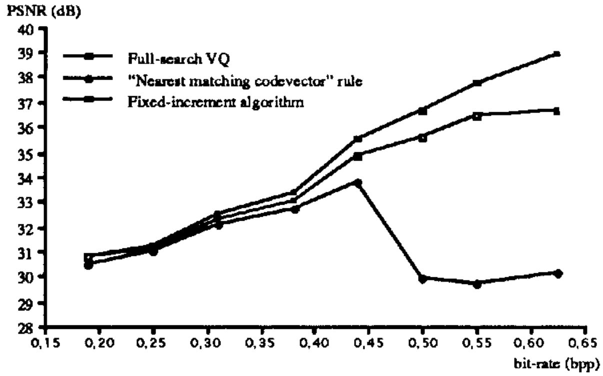

Figure 1: Fixed–increment algorithm/"nearest matching codevector" rule, codebook initilization random.

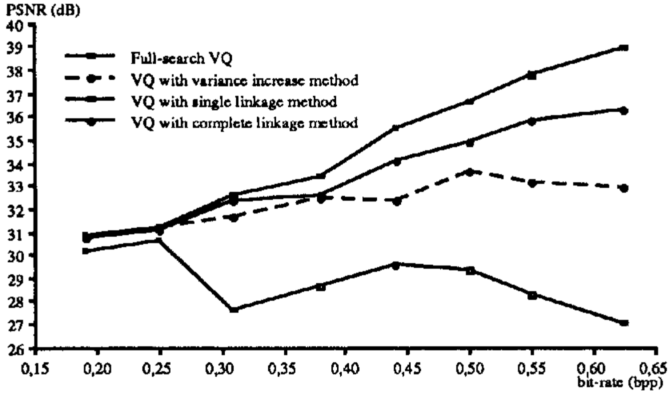

Figure 2: VQ performances, codebook initilization random.

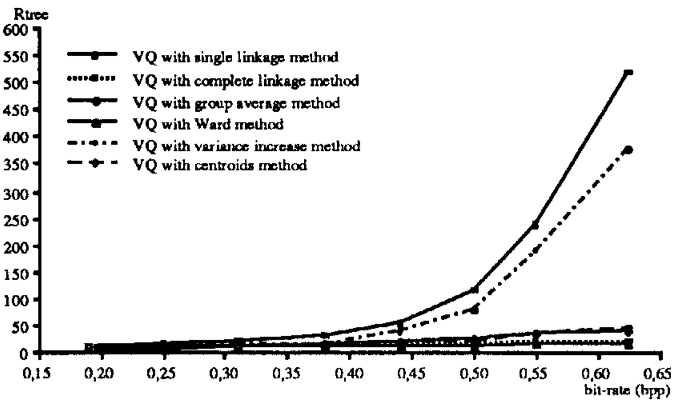

Figure 3: Compactness and chaining effects, codebook initilization random.

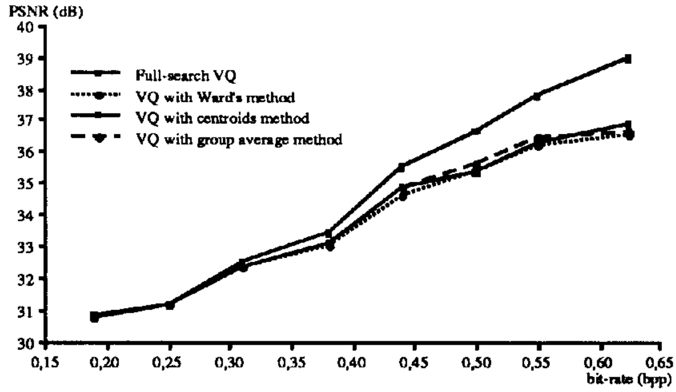

Figure 4: VQ performances, codebook initilization random.

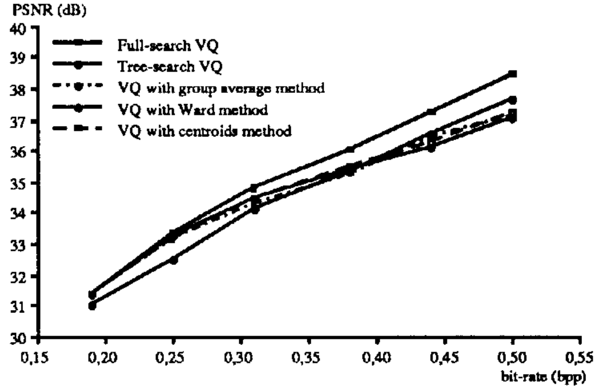

Figure 5: SAHN/TSVQ performances, "splitting" initilization of the codebook.

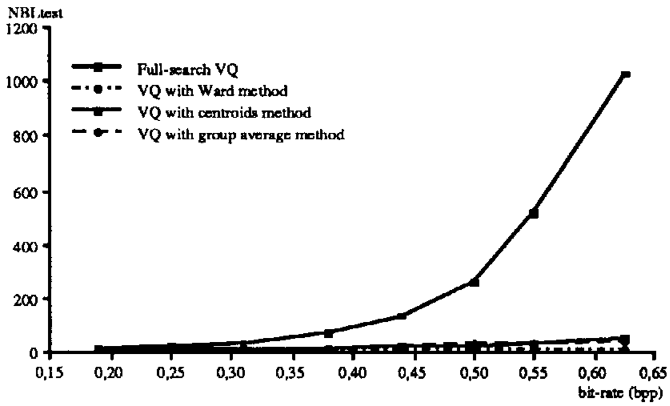

Figure 6: VQ complexity, codebook initilization random.