

# Tree-Structured Lattice Vector Quantization for Video Coding

Vincent Ricordel, Claude Labit and Moncef Gabbouj

*Abstract*—The purpose of this paper is to introduce a new vector quantizer (VQ) for the compression of digital image sequence. The proposed approach unifies both efficient coding methods: a fast lattice encoding and an unbalanced tree-structured codebook design according to a distortion *v.s.* rate tradeoff. This tree-structured lattice VQ (TSLVQ) is based on the hierarchical packing of embedded truncated lattices. In the paper, we investigate its complete design with: the lattice truncation, the multi-stage procedure of quantization, the unbalanced tree-structured design and the determination of the best lattice respectively to this method. Experimental results are given with the TSLVQ taking place in sub-band coders: the detection and the processing of the outlying input vectors are then defined, as well as a bit allocation strategy. The TSLVQ offers some original solutions to usual lattice VQ drawbacks, with a space partition according to the source distribution, and a simple labeling of the lattice points. The codebook automatically obtained is well suited for prediction error coding, and a fast updating of the code-vectors can be performed.

## I. INTRODUCTION

The most promising method introduced for the VQ<sup>1</sup> [1], [2], is certainly lattice vector quantization (LVQ) [3], [4], [5], for which the codebook is not calculated. It is defined as a particular subset of regularly arranged points in an  $n$ -dimensional space, centered in zero (lattice [6], [7]). When designing an LVQ, the difficulty is not the same as an LBG-type algorithm [8] which as computationally expensive encoding and codebook storage, but lies with the choice of lattice, its truncation and labeling of the remaining points. In this paper, motivated by video coding applications, we develop a new VQ for which lattice use is simplified. The paper is organized as follows. Brief reviews on VQ and LVQ are presented respectively in Section II and Section III. The source characteristics are given in Section IV in order to specify the adapted quantizer. Section V deals with the TSLVQ design. Finally, Section V also presents experimental results at a low bit rate.

## II. VECTOR QUANTIZATION

VQ has been investigated extensively in recent years [9], [2]. Let  $x(n)$ :  $\mathbf{x} = (x(1), \dots, x(k))$  be a  $k$ -component source vector with joint probability density function (pdf)  $p_{\mathbf{X}}(\mathbf{x})$ . A VQ of dimension  $k$  and size  $L$  is defined as a

function that maps a vector  $\mathbf{x}$  into one of  $L$  reproduction vectors  $\mathbf{y}_1, \dots, \mathbf{y}_L$  belonging to  $\mathbb{R}^k$ . We have:

$$\begin{aligned} Q &: \mathbb{R}^k \longrightarrow \mathcal{D} \\ \mathbf{x} &\longmapsto Q(\mathbf{x}) = \mathbf{y}_i \end{aligned}$$

Where  $\mathcal{D}$ , the set of reproduction vectors or codewords, is the codebook. This chart implicitly determines a partition of  $\mathbb{R}^k$  in  $L$  non-overlapping Voronoï regions  $C_i$  defined by the equation (where  $L_p(\mathbf{x}) = \sum_{i=1}^k |x_i|^p$  is the norm):

$$C_i = \left\{ \begin{array}{l} \mathbf{x} \in \mathbb{R}^k / Q(\mathbf{x}) = \mathbf{y}_i, \\ \text{if } L_2(\mathbf{x} - \mathbf{y}_i) \leq L_2(\mathbf{x} - \mathbf{y}_j), \forall j \neq i \end{array} \right\}$$

The total distortion [per dimension] is given by:

$$D = \frac{1}{k} E\{L_2(\mathbf{X} - Q(\mathbf{X}))\} = \frac{1}{k} \sum_{i=1}^L \int_{C_i} L_2(\mathbf{x} - \mathbf{y}_i) \cdot p_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}$$

In this transmission and storage context, a binary word or index  $c_i$  of length  $b_i$  bits is assigned to each codeword  $\mathbf{y}_i$ . Thus a VQ combines both functions: an encoder that, from the input vector  $\mathbf{x}$ , generates the index  $i$  specified by  $Q(\mathbf{x})$ ; a decoder that, from this index and by listing  $\mathcal{D}$ , generates the corresponding reproduction vector  $\mathbf{y}_i$ . The average binary word length is given by the entropy measure of the codebook (where  $p_i$  is the occurrence probability of  $\mathbf{y}_i$ ):

$$R \simeq H(\mathcal{D}) = -\frac{1}{k} \sum_{i=1}^L p_i \cdot \log_2(p_i) \quad [\text{bit/dimension}]$$

An adaptive entropy coder [10] is supposed to produce this minimum bit rate necessary to achieve the distortion  $D$ . In fact a codebook design with an entropy constraint has to be used. The ECVQ algorithm [11] is then optimal, but this generalization of the LBG algorithm [8] is very computationally expensive.

## III. LATTICE VECTOR QUANTIZATION

The LVQ [3], [12], [4] has been successfully introduced in order to overcome the LBG-type algorithm [8] drawbacks. The encoding, based on rounding and scaling operations, is simple and independent of the codebook size. There is no need to seek among all the reproduction vectors, and practically no norms are computed. Because of the predefined structure of the lattice, there is no need to transmit the codebook and no training procedure is required to design it. But a lattice can only optimally quantize uniform source

V. Ricordel, SIS/ISITV, Av. Pompidou BP56, 83162 La Valette Cedex, France. E-mail: ricordel@univ-tln.fr .

C. Labit, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France. E-mail: labit@irisa.fr .

M. Gabbouj, DMI/SPL, TUT, P.O. Box 553, 33101-Tampere, Finland. E-mail: gabbouj@cs.tut.fi .

<sup>1</sup>From now VQ means vector quantizer as well as vector quantization.

and, because of its infinite size, it must be truncated to index the codewords. Thus the LVQ of a non-uniform source becomes complex. We distinguish the different steps:

1. the lattice choice which defines the “space filling advantage” [13], [14], [15] of the LVQ. Some best lattices are known for the dimensions 2, 4, 8, 16 and 24, because they offer the smallest distortion when quantizing an uniform source [6], [7]. But in practice, the fast quantization algorithms [16] associated with the lattices  $\mathbb{Z}^k$ ,  $D_k$ ,  $E_8$  and  $A_{16}$ , are the main criterion for the selection;

2. the determination of the codebook shape. When truncating the lattice, the aim is to map the most probable distribution of the vector source, in order to exploit the “shape advantage” [15] of the VQ. The number of the remaining lattice points in the codebook depends on the allocated rate;

3. the source normalization. The fast quantization algorithm [16] is used, once the source normalization has been achieved. We can describe this operation as a projection of the input vectors into the truncated lattice volume. A scaling function [17], [18], [5], [19], [20] (linear or not) takes place before the quantization module. A particular process for the outlying source vectors (which still don’t belong to the codebook after normalization) must be defined [21], [4], [22], [23];

4. the labeling of the lattice codebook points. If the encoding step is simpler with a lattice, the last step of indexing the codewords becomes complicated when the codebook size is large. The index must be calculated [24], [3], [22], [23], [25], [20], and the methods aim to realize the best tradeoff between the calculation cost and the conversion table storage. If no training procedure has been required yet, it is usually used in order to carry out the entropic index associated with the lattice points (to get their occurrence probability).

The inverse quantization procedure consists in decoding the received index, finding the corresponding lattice point and re-normalizing it.

The choice of the metric  $L_2$  [26], [18], [4] (respectively  $L_1$  [3], [18], [21], [4], [22]) permits to shape the codebook into a hyper-sphere (respectively an hyper-pyramid) and to count the points. As a result the basic LVQ is only adapted for symmetric and Gaussian (respectively Laplacian) source distributions which map such a codebook. This restrictive modelization of the source offers some accurate and sophisticated methods to achieve the LVQ design, but the well-perform condition collapses considering a complex source coding at low bit rate. With the TSLVQ, based on an embedding lattice strategy, simple procedures are implemented to overcome these drawbacks.

#### IV. SPECIFICATION OF THE VECTOR SOURCE

Predictive coding and subband coding [27], [28], [29], [30], [31] are unavoidable decorrelation methods for the compression of digital image sequence. Because an hybrid coder aims to apply precisely two main rules: the non-transmission of the predictable information and, the non-transmission of the no-perceivable information by the

human visual system (HVS).

Fig 1 shows the generic coder for which our VQ is devoted. Note that we don’t study in the paper the quantization aspect of the information performed by the motion estimation [31] between each image pair of the input sequence, but the VQ of the transformed prediction errors. The current prediction error image is the difference between the prediction image, obtained by a motion compensation of the previous decoded image, and the input image. The prediction error image is then transformed and decomposed in subimages or subbands before the VQ.

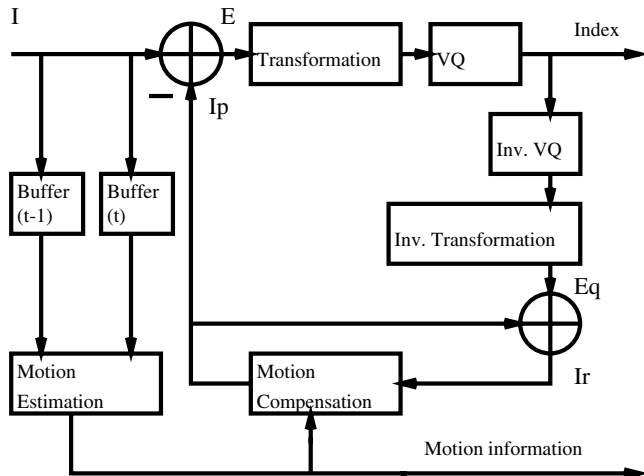


Fig. 1. Hybrid coder with forward motion compensation. I: input image from the sequence; Ip: prediction image; Ir: reconstructed image; E: prediction error image; Eq: quantized prediction error image.

The inter-intra decorrelation steps shape the signal before its quantization. Because in each subimage the data are classified according to their frequential orientation and resolution. A separate VQ of each subimage [4] is well-suited in order to exploit the dependencies between the transformed coefficients, and the bit allocation is performed, such as, taking account of the HVS propriety, the low frequency subimages receive more bits (but the resulting gain is subjective).

The monodimensional pdf of such a subimage signal is commonly mapped by a centered Generalized Gaussian function whose narrowest highlights the correlation between the coefficients [4]. For the predictive sources, the multidimensional repartition of correlated vectors is made into some ellipses oriented according to the bisector axis [32]. Because the edges within the differential images are characterized by some pixels with a very near absolute value and an opposite sign (see figure 2), and the corresponding transformed coefficients conserve this propriety.

However the shaped signal obtained after the hybrid decorrelation chain is always non-stationary [27], [33]. Only an adaptive VQ [34], [35], [36] is capable of adapting to changing source statistics as the coding progresses. Such as, from a source representative training sequence, a very fast updating of the VQ codebook is achieved (see figure 3).

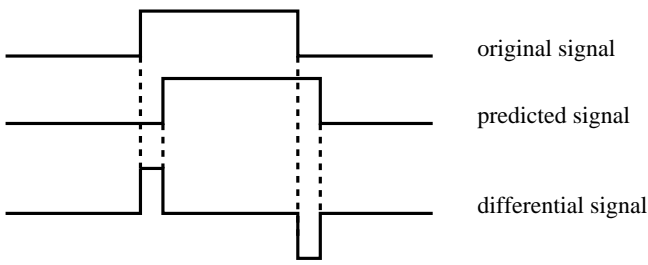


Fig. 2. The propriety of the differential signal edges.

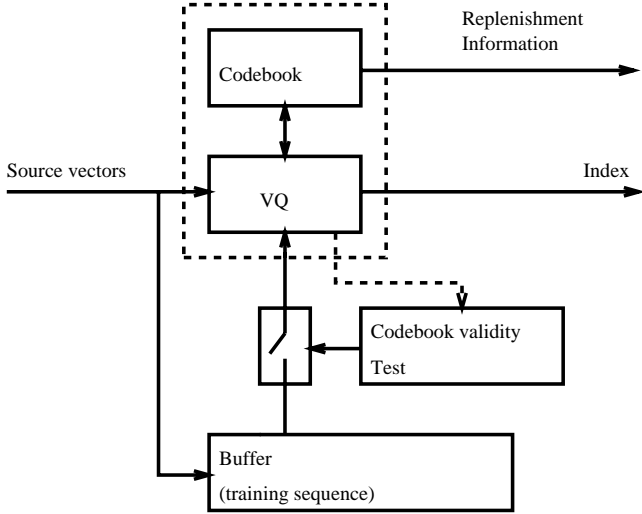


Fig. 3. Adaptive VQ scheme. The codebook validity test consists in a mean distortion-based criterion.

## V. TREE-STRUCTURE LATTICE VQ DESIGN

In this chapter the different steps of the TSLVQ design are progressively described in order to justify our choices.

### A. Hierarchical Set of Embedded Lattices

With respect to the vector space dimension, the lattice is chosen among  $\mathbb{Z}^2$ ,  $D_4$  or  $E_8$ , for which Conway and Sloane determined fast quantizing and decoding algorithms [16]. This support lattice is then truncated such as it can be embedded, after contraction, into its central Voronoï region. The embedding is optimal if the Voronoï cells of the embedded lattice cover exactly the receiving Voronoï region. But this result depends on the geometric propriety of the support lattice. Our goal becomes to achieve a sub-optimal embedding such as the receiving Voronoï region is full of a maximum number of complete Voronoï cells of the support lattice. In practice, the embedding consists in shifting the scale of the lattice to embed with respect to the support lattice scale that is fixed and equal to one. Reciprocally, in order to solve the problem, we consider that the support lattice scale is fixed, then the central Voronoï cell of the receiving lattice is dilated.

Let us describe a lattice like a packing of identical spheres in  $\mathbb{R}^k$  where the lattice points are the sphere centers [6], [7]. The sphere radius  $\rho$  is called the packing radius. We call the touch points, the points between the packed spheres. Note that the Voronoï cell edges are the hyperplanes tan-

gent to the spheres in these touch points. So (see figure 4) <sup>2</sup> if  $\rho$  is the packing radius of the support lattice, the packing radius of the receiving Voronoï region is then:

$$b \times \rho \quad / \quad b \in \mathbb{R} \text{ and } b > 1$$

A sub-optimal embedding is then achieved if we dilate the receiving Voronoï cell by the scaling factor:

$$b = 2 \times n + 1 \quad / \quad n \in \mathbb{N}^*$$

In order that the touch points of the dilating lattice correspond to some touch points of the support lattice.

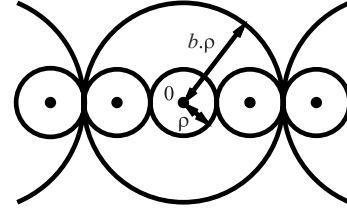


Fig. 4. Embedding principle. The scaling factor is here:  $b = 3$ .

As a result (see figures 5 and 6) a maximal number of edges of the support lattice Voronoï cells are merged into the edges of the scaled Voronoï cell.

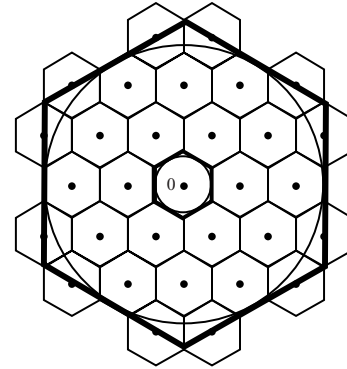


Fig. 5. A sub-optimal embedding with the hexagonal lattice ( $b = 5$ ).

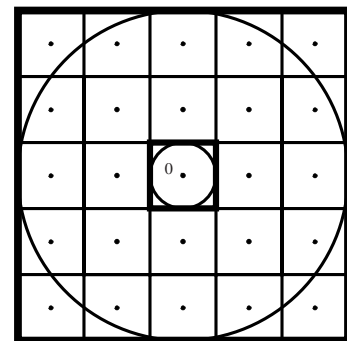


Fig. 6. An optimal embedding with the squared lattice ( $b = 5$ ).

<sup>2</sup>For simplicity the figures are in dimension 2 and often with the squared lattice  $\mathbb{Z}^2$ , but the results can be generalized to higher dimensions with the other lattices.

The support lattice is truncated because only its Voronoï cells, completely or partially within the dilated Voronoï cell are conserved.

The hierarchical set of embedded lattices is achieved such as it is possible to embed a lower scale truncated lattice into a cell of the next higher scale truncated lattice. So the scaling factor between two consecutive lattices of the hierarchy is  $b$  (see figure 7).

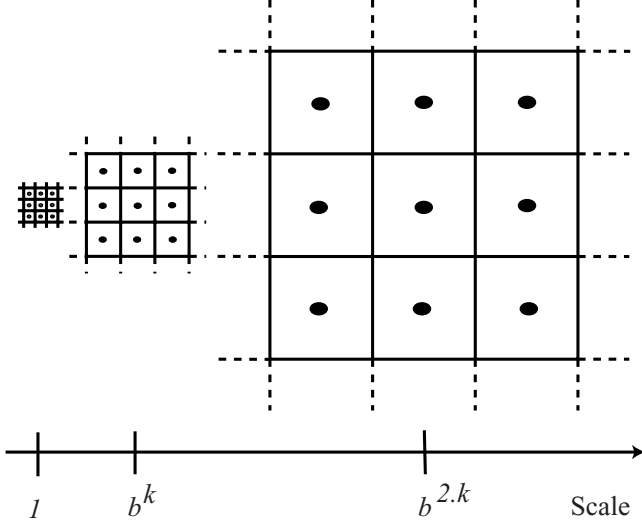


Fig. 7. Hierarchical set corresponding to the squared lattice ( $b = 3$ ).

### B. Quantization Procedure

The principles of the quantization are:

1. a source vector is projected into a first truncated lattice;
2. to get a finer quantization, another lower scale truncated lattice is embedded into the Voronoï cell where lies the input vector;
3. the previous operation can be repeated.

Obviously it is more convenient to deal with the input vector scale than to use several lattices with different scales. The figure 8 illustrates the resulting multi-stage quantization procedure using successive scaling and translating operators. We have:

- the scaling factor used to project the input vector  $\mathbf{x}$  into the first truncated lattice:

$$F = \frac{b \times \rho}{\sqrt{L_{2 \max}}} \quad (1)$$

where  $L_{2 \max}$  is the maximal  $L_2$  norm of  $\mathbf{x}$ , this constant energy can be estimated from a training sequence. So all the inputs are projected into an hyper-sphere whose radius equals  $(b \times \rho)$  because:

$$\begin{aligned} \sum_{i=1}^k (F \times x_i)^2 &= F^2 \times \sum_{i=1}^k x_i^2 = F^2 \times L_2(\mathbf{x}) \\ &= (b \times \rho)^2 \times \frac{L_2(\mathbf{x})}{L_{2 \max}} \leq (b \times \rho)^2 \end{aligned}$$

- in the normalized space, the scaling factor used to project each translated vector into the next truncated lattice of the hierarchy:  $b$
- the reproduction vector of the truncated lattice for the  $j$ -th stage:  $\mathbf{y}_j$

The final value of the reproduction vector associated with  $\mathbf{x}$  will be:

$$\mathbf{y} = \frac{1}{F} \times \sum_j \frac{\mathbf{y}_j}{b^{j-1}}$$

where  $j$  points out the number of the stages. Note that at each step the same fast quantization algorithm is used.

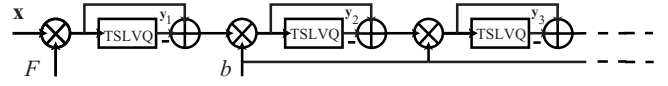


Fig. 8. Quantization scheme.

### C. Tree-Structured Codebook

The codebook has a  $B$ -ary tree structure [2], [19], [37], [38] where  $B$  corresponds to the number of points of the basic truncated lattice. Each lattice point (*i.e.* each reproduction vector) and its Voronoï cell are associated with a tree node. The 0 vector and the whole source space are associated with the root tree. The possible children of a node are the points of the lattice which is embedded into the parent Voronoï cell. A stage in the tree corresponds with a scale in the lattice hierarchy: the deeper is the tree, the finer is the resolution. The final codebook is the set of terminal nodes or leaves.

Tree-structured codebook designs are illustrated by the figure 9. These quantization of synthetic sources shows how the space partition matches with the source distribution. A progressive splitting of the vector space is achieved if, when embedding a truncated lattice, the number of the new reproduction points are restricted. So the scale factor  $b$  is fixed at its minimal value:

$$b = b_{\min} = 3$$

The tree-structured codebooks of the figure 9 are balanced, namely all the leaves are at the same stage. A more efficient method consists of splitting the vector space according to a distortion *v.s* rate criterion, the tree-structure codebooks are then unbalanced.

### D. Unbalanced Tree-Structured Codebook

Two classical strategies have been explored in order to design an unbalanced tree according to a distortion *v.s* rate tradeoff: a tree pruning [39], [40] or a tree growing approach [41], [42], [43]. It aims to split the vector space region where the distortion is high, and to avoid simultaneously a prohibitive cost in rate.

A training procedure is performed to design the tree-structured codebook. Let  $\mathbf{y}_{n_i}$  and  $C_{n_i}$ , the reproduction vector and the Voronoï cell associated with the node  $n_i$ ,  $N$  the training sequence size (*i.e.* the source vector number) and  $\text{card}(C_{n_i})$  the number of input vectors that lie within  $C_{n_i}$ . Then, each tree node is characterized by <sup>3</sup>:

<sup>3</sup>From now we use "distortion" instead of "average distortion", and "rate" instead of "average entropy code length".

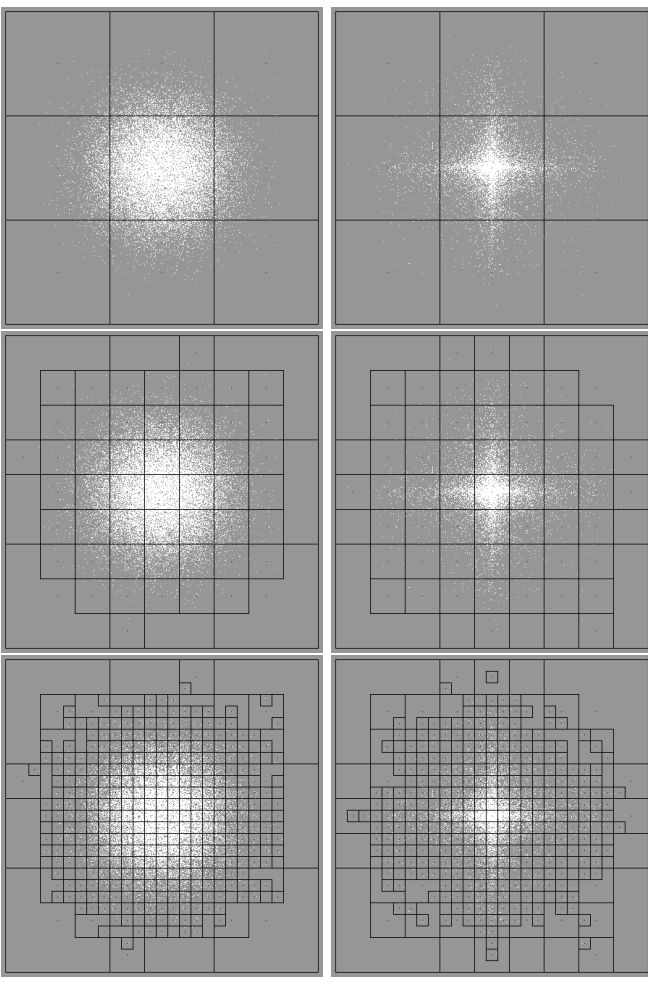


Fig. 9. Tree-structured codebook designs using  $\mathbb{Z}^2$  lattice ( $b = 3$ ). The source vectors are the white dots, the distribution of their coordinates is i.i.d and respectively Gaussian (on the first column), and Generalized Gaussian (second column). A reproduction vector (in black) and its Voronoi cell are depicted only if they are used for the quantization. Three successive quantization stages are illustrated, one per line.

- an occurrence probability:

$$P(n_i) = \frac{\text{card}(C_{n_i})}{N}$$

- a distortion:

$$d(n_i) = \frac{1}{\text{card}(C_{n_i})} \times \sum_{\mathbf{x} \in C_{n_i}} L_2(\mathbf{x} - \mathbf{y}_{n_i})$$

- a rate (assuming that an entropy encoding is used for the leave indexing):

$$l(n_i) = -\log_2 P(n_i)$$

Consequently we get for a subtree  $\mathcal{S}$  ( $\tilde{\mathcal{S}}$  symbolises the set of its leaves):

- a distortion:

$$d(\mathcal{S}) = \sum_{n_u \in \tilde{\mathcal{S}}} P(n_u) \times d(n_u)$$

- a rate:

$$l(\mathcal{S}) = \sum_{n_u \in \tilde{\mathcal{S}}} P(n_u) \times l(n_u)$$

The tree pruning approach is known as the BFOS algorithm [39]. First a complete tree  $\mathcal{T}$  is achieved, it will be successively pruned according to the BFOS criterion. If a branch  $\mathcal{S}_{n_i}$  (*i.e* the subtree rooted at  $n_i$ , and such as  $\tilde{\mathcal{S}}_{n_i} \subset \tilde{\mathcal{T}}$ ) is removed, we get:

- an increase in distortion:

$$\Delta d(\mathcal{S}_{n_i}) = P(n_i) \times d(n_i) - d(\mathcal{S}_{n_i})$$

- a decrease in rate:

$$\Delta l(\mathcal{S}_{n_i}) = l(\mathcal{S}_{n_i}) - P(n_i) \times l(n_i)$$

- the BFOS criterion:

$$\lambda(n_i) = \frac{\Delta d(\mathcal{S}_{n_i})}{\Delta l(\mathcal{S}_{n_i})}$$

$\lambda(n_i)$  can be interpreted as a possible piece-wise of the slope of the experimental distortion *v.s* rate curve. Thus we successively prune the branches for which the ratio  $\lambda(n_i)$  is minimal such as the total distortion is increased a little as possible for a decrease in total rate. The resulting sequence of distortion *v.s* rate pairs corresponding to the pruned subtrees  $\mathcal{S}$  (*i.e* a subtree with the same root as the full tree  $\mathcal{T}$ ) lies on the convex hull of the operational distortion *v.s* rate function. The BFOS algorithm could be considered as using a Lagrangian approach that aims to minimize the functional  $J(\mathcal{S}) = d(\mathcal{S}) + \lambda \times l(\mathcal{S})$  where  $\lambda$  is interpreted as a Lagrangian multiplier [39], [44]: varying  $\lambda$ , all the pairs on the convex hull of the distortion *v.s* rate function can be reached. The distortion or the rate associated to the pruned subtree enables to interrupt this codebook design procedure as soon as a threshold is exceeded. Because of the storage complexity of the first complete tree, the tree pruning approach is not suitable for the TSLVQ whose tree arary number is high.

As a starting point of the tree growing approach [42], [2], any tree structure can be used. Typically we start with the root, afterward an individual leaf splitting is applied to grow the tree. Exactly for each loop of the growing process we split all the tree leaves, but only the new branch with the best distortion *v.s* rate tradeoff is conserved. Let be  $\mathcal{S}^j$  the subtree obtained at the end of the  $(j - 1)$ -th loop (at the initialisation  $\mathcal{S}^0 = n_0$ ), its leaves are the  $n_i$ . For the next loop, when splitting the  $n_i$  we get the new branches  $\mathcal{S}_{n_i}^j$  with for each:

- a decrease in distortion:

$$\Delta d(\mathcal{S}_{n_i}) = P(n_i) \times d(n_i) - d(\mathcal{S}_{n_i})$$

- an increase in rate:

$$\Delta l(\mathcal{S}_{n_i}) = l(\mathcal{S}_{n_i}) - P(n_i) \times l(n_i)$$

- the ratio:

$$\lambda(n_i) = \frac{\Delta d(\mathcal{S}_{n_i})}{\Delta l(\mathcal{S}_{n_i})}$$

So now we conserved the branch for which  $\lambda(n_i)$  is maximal. The distortion and the rate associated with the new pruned subtree are:

$$d(\mathcal{S}^j) = d(\mathcal{S}^{j-1}) - \Delta d(\mathcal{S}_{n_i}^j)$$

$$l(\mathcal{S}^j) = l(\mathcal{S}^{j-1}) + \Delta l(\mathcal{S}_{n_i}^j)$$

The growing process will stop as soon as a threshold in distortion or in rate is exceeded. This local approach is adapted to the TSLVQ because the amount of stored information fits exactly to the codebook size that grows progressively.

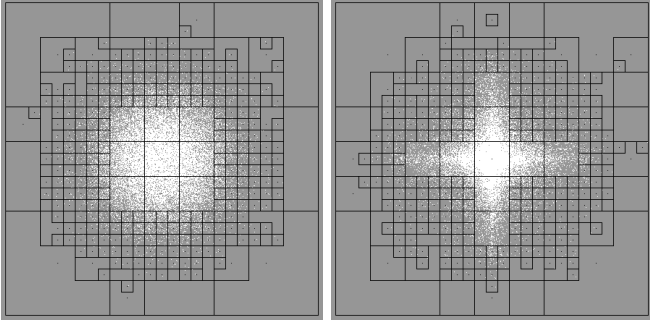


Fig. 10. Greedy tree approach illustrations considering the TSLVQ of the two synthetic sources of the figure 9 and using  $\mathbb{Z}^2$  lattice ( $b = 3$ ). The training ratio (*i.e* the ratio of the training sequence size on the number of the tree leaves) is over than 150.

The figure 10 shows the pruned trees corresponding to the TSLVQ of the synthetic sources of the figures 9. These examples illustrates how the TSLVQ is adapted to differential or hybrid image source coding. Because, for a given rate, the high-density space region where are located the lowest error magnitudes is coarsely quantized (a "dead zone" [30] appears) in order to permit a finer coding of the low density region where lie relevant input vectors.

#### E. Labeling of the lattice points

The TSLVQ is a multi-stage VQ [2] where the same LVQ is used at each stage. Therefore it's convenient to achieve out-line a look-up table containing the index of the basic truncated lattice points. This look-up table is supposed to be known by the coder as well as the decoder, then no representation vector have to be transmitted. The tree structure of the TSLVQ codebook permits its compact presentation, we proceed in two tree scans:

- the first is achieved just to give a number to the nodes;
- the second permits to store for each node: its children and parent numbers, the index of the corresponding lattice point (this index belongs to the previous look-up table) and, only for a leaf, its entropy code word.

This TSLVQ codebook presentation will be upgrade for the adaptive quantization.

#### F. Determination of the best lattice

The goal is to determine the best lattice for the TSLVQ among the lattices for which Conway and Sloane determined fast quantizing and decoding algorithm [16]. Figure 11 shows experimental codebook entropy *v.s* distortion

curves where we compare  $\mathbb{Z}^4$  with  $D_4$ , and  $\mathbb{Z}^8$  with  $E_8$  (note that  $D_4$  and  $E_8$  are the best quantizing lattice respectively to their dimension and considering the high resolution theory [6], [7]). It appears that  $\mathbb{Z}^k$  performs better than the others: lower distortion and rate are obtained.

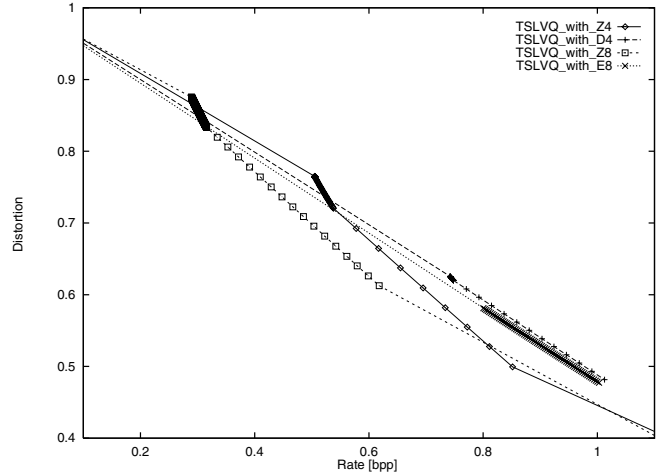


Fig. 11. TSLVQ using a tree growing approach: experimental distortion *v.s* rate curves with an *i.i.d* synthetic Gaussian source (source variance  $\sigma^2 = 1$ ). The training ratio is over than 150.

$\mathbb{Z}^k$  is the best lattice for the TSLVQ because it is the only lattice that produces an optimal embedding: the cubic truncated lattice recovers exactly, when we embed it, the cubic Voronoï cell. Figure 12 illustrates simply the optimal embedding advantage. Suppose that the source vectors are uniformly distributed into the shaded region. For the truncated cubic lattice the decrease in distortion and the increase in rate are equally shared between all the nine points. For the hexagonal lattice (which is the best quantizing lattice for the dimension 2 [6], [7]) the six peripheral points are less probable, it induces an increase in rate and a less decrease in distortion.

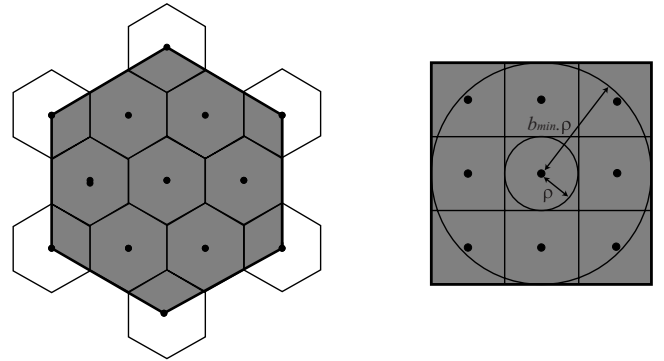


Fig. 12. An optimal embedding with the cubic lattice and a sub-optimal with the hexagonal lattice ( $b = 3$ ).

Using  $\mathbb{Z}^k$  implies that the resulting TSLVQ has not a space filling advantage, but in practice this gain is very low [15]. On the other hand  $\mathbb{Z}^k$  is the least dense lattice, so the truncated lattice point number (*i.e* the tree nodes) is reduced involving a more accurate vector space partition.

### G. Processing of the outlying input vectors

A training procedure is used in order to set up the TSLVQ codebook. After that, during the coding, the input vectors will not be exactly the training sequence vectors. So we must detect and process separately the probable outlying vectors (*i.e* the source vectors that do not belong to the codebook after normalisation).

Because the truncated lattice is an hyper-cube, the  $L_\infty$  norm of the vectors  $\mathbf{u}$  within is such as (see fig 12):

$$L_\infty(\mathbf{u}) = \max_{i=1,\dots,k} |u_i| \leq (b_{min} \times \rho)$$

The scaling factor (see equation 1) that aims to project the input vectors  $\mathbf{x}$  into the hyper-cube equals:

$$F = \frac{b_{min} \times \rho}{\sqrt{L_{2\ max}}}$$

As a result, the source vectors  $\mathbf{x}$  that can be effectively quantized by the codebook have to verify:

$$L_\infty(\mathbf{x}) = \max_{i=1,\dots,k} |x_i| \leq \sqrt{L_{2\ max}}$$

This test using the  $L_\infty$  norm is obviously very fast. The detected outlying vectors are projected on the frontier of the first hyper-cube by using:

$$\text{If } |x_i|_{i=1,\dots,k} > \sqrt{L_{2\ max}} \implies x_i = \text{sign}(x_i) \times \sqrt{L_{2\ max}}$$

These vectors are then quantized. They will be decoded as usual. The resulting overload distortion can be large but the outlying vector probability is low [26].

### H. Bit allocation

The allocation problem occurs because we have to share the bit ressource between the subbands [2], [45], [46]. It consists in minimizing the global distortion  $D$  subject to the constraint that the global rate  $R$  is under a threshold  $R_d$ .

For each subband  $j$  a separate TSLVQ is set up. Each pruned TSLVQ configuration is numbered by  $i$ . The set of potential quantizers  $q_{i,j}$  of the  $j$ -th subband is the set of the corresponding pruned TSLVQ configurations.  $d_{i,j}$  and  $r_{i,j}$  are the distortion and the rate when coding the subband  $j$  with  $q_{i,j}$ . The transformation is supposed orthogonal and there are  $M$  subbands, so the problem is:

$$\min D = \min \sum_{j=0}^{M-1} d_{j,i} \quad \text{subject to} \quad R = \sum_{j=0}^{M-1} r_{j,i} \leq R_d$$

A combination of  $M$  quantizers (one for each subband) involves a point  $(R, D)$  in the distortion *v.s* rate space, all the combinations produce a cluster (see figure 13 as an example). The problem becomes the determination on the cluster convex hull of the point whose rate is just lower than  $R_d$ . In order to reduce the amount of calculus the Lagrange-multiplier method is introduced to solve:

$$\min(D + \lambda \times R) \iff \sum_{j=0}^{M-1} \min_{q_{i,j}} (d_{i,j} + \lambda \times r_{i,j})$$

The complexity decreases because the reduction of distortion is now achieved separately from each subband. The general form of the algorithm is:

- 1 The convex hull for each subband is calculated. With the TSLVQ we obtain it directly when growing the tree, by storing the distortion *v.s* rate point corresponding to each new branch addition;
- 2 The point on the global convex hull is determined, its rate is just below  $R_d$ .

For this second step, the algorithmic solution proposed by Shoham en Gersho [45] is particularly adapted. It is based on the calculation of singular values of the Lagrange-multiplier  $\lambda$ . Precisely, a singular value of  $\lambda$  is the slope of the line that pass through two consecutive points of the convex hull. So from a first point on the hull, by successive calculations of singular values, we get the global convex hull.

In practice with the TSLVQ, a first (respectively second) point is  $(R, D)$  where  $D$  equals to the sum of the distortions when quantizing each subband with the minimal (respectively maximal) rate allowed. The successive singular values of  $\lambda$  are obtained by ordering in an increasing (respectively decreasing) manner the BFOS criteria [39] and considering all the subbands. The figure 13 shows an experimental result. A drawback appears because there are some large gaps between some points of the global convex hull.

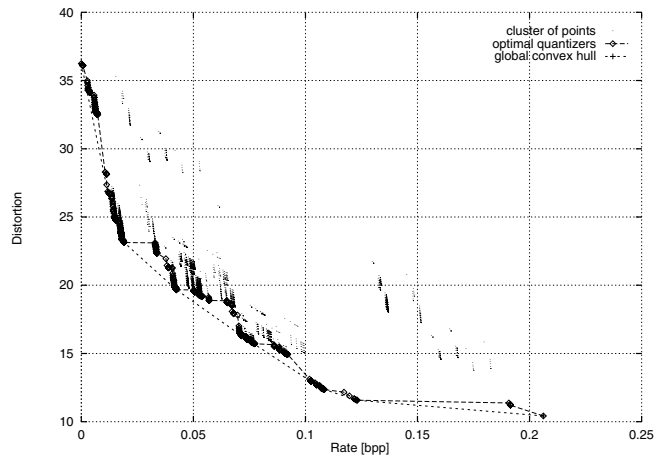


Fig. 13. Cluster of  $(R, D)$  points, convex hull and optimal quantizers.

We want to improve the process in order to get the optimal quantizers namely the points just above the convex hull. We proceed in two steps:

- 1 The previous method permits to get two points of the convex hull that surround the rate target  $R_d$ ;
- 2 From these two points the Shoham's algorithm [45] is used again in order to get a local portion of the convex hull.

The curve titled "intermediary quantizers" on the figure 14 is achieved by calculating the portions of convex hulls between each consecutive couples of points on the global convex hull. As a result a lot of optimal quantizers are reached.

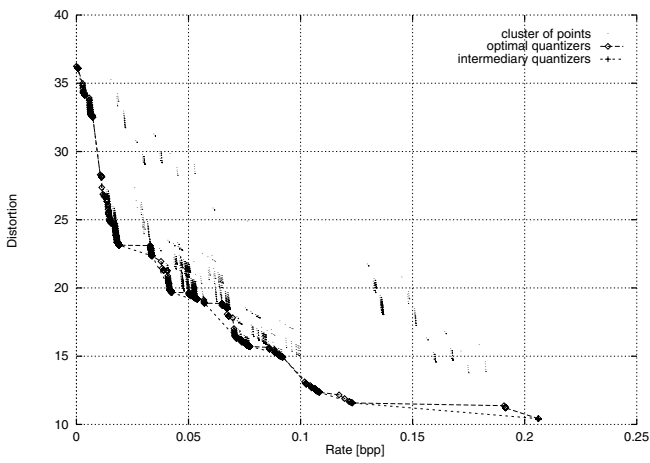


Fig. 14. Cluster of  $(R, D)$  points, convex hull and intermediary quantizers.

### I. Adaptive Vector Quantization

For the adaptive TSLVQ [34] the codebook is set up in two parts: a stump from several types of training sequences, this structure is achieved outline; some branches added according to the image sequence to be quantized, these branches constitute the only transmitted information to update the codebook. Because of the binary allocation that implies a tree pruning, there are precisely four steps in the process (see the figure 15):

- 1 Construction of a large stump,
  - 2 Stump binary allocation, the chosen rate threshold is very low,
  - 3 Addition of large branches,
  - 4 Codebook final binary allocation with the desired rate.
- In order to update the codebook, only the step 3 and 4 are carried out.

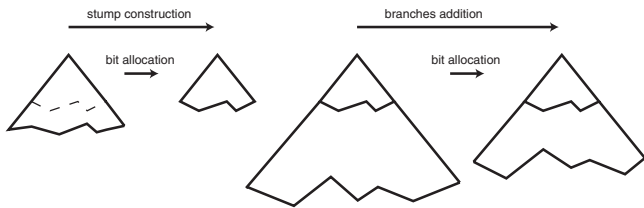


Fig. 15. The four steps of the adaptive TSLVQ codebook.

## VI. EXPERIMENTAL RESULTS

The TSLVQ takes place in a subband coder (see figure 1) devoted to very low bit rate coding of video-phone image sequence. This coder is MPEG based because the tools are a block matching for the motion estimation/compensation and a DCT for the transform. There are four experimental stages:

- 1 the training sequences constitutions with the coder in an open loop,
- 2 the subband codebook design,
- 3 the bit allocation between the subbands,
- 4 the image sequence coding with the coder in closed loop.

For the numerical results, in addition to the Peak Signal to Noise Ratio (PSNR), the prediction gain  $G_p$  and the quantization gain  $G_q$  are distinguished,  $PSNR = G_p + G_q$  with:

$$G_p = 10 \times \log_{10} \frac{255^2}{\frac{1}{S} \times \sum_{i=1}^S e_i^2}$$

$$G_q = 10 \times \log_{10} \frac{\sum_{i=1}^S e_i^2}{\sum_{i=1}^S (e_{qi} - e_i)^2}$$

where  $S$  is the image size,  $e_i$  the prediction errors and  $e_{qi}$  the vector quantized prediction errors. The simulations are made using QCIF image sequences (images extracted from the sequences are shown on figure 16): "Miss America", "Salesman", "Carphone" and "Mother-and-daughter". For the CPU time, the target computer is a Sparc-Station 20 (75 Mhz).



Fig. 16. Images extracted from the original sequences.

### A. Results with classical TSLVQ

#### A.1 Codebook design

The block size is  $(2 \times 2)$  so 4 sub-bands are set up by putting together the coefficients relating to the same frequency activity. For each sub-band a codebook is constructed as shown in the figure 17: the vector shape is adapted in order to exploit the redundancy and the indicated rates are the thresholds before bit allocation.



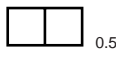
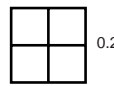
	1 bpp		0.5 bpp
A		B	
	0.5 bpp		0.2 bpp
C		D	

Fig. 17. Codebook: vector shapes, sub-band labels and maximal entropies (before bit allocation).

Images from all four different sequences are used for the codebook design, table I shows the numerical results before bit allocation and table II after bit allocation. Note that as the vector dimension increases the number of code-vectors increases too, so the training sequence size must be large in order to ensure a high enough training ratio. The CPU times for these codebook constructions are small although the programming is not optimal.



subband label	training sequence	CPU time	codeword number	codebook entropy
A	4 images	4.93 s	55	0.698 bpp
B	12 images	6.15 s	106	0.213 bpp
C	12 images	6.22 s	133	0.231 bpp
D	200 images	69.43 s	1982	0.083 bpp

TABLE I

CLASSICAL TSLVQ, CODEBOOK DESIGN FROM THE FOUR SEQUENCES: BEFORE BIT ALLOCATION.

subband label	threshold: $R_d = 0.2$ bpp final rate: $R = 0.195$ bpp		threshold: $R_d = 0.1$ bpp final rate: $R = 0.096$ bpp	
	codeword number	codebook entropy	codeword number	codebook entropy
A	20	0.332 bpp	20	0.332 bpp
B	106	0.213 bpp	26	0.020 bpp
C	133	0.231 bpp	34	0.027 bpp
D	92	0.003 bpp	92	0.003 bpp

TABLE II

CLASSICAL TSLVQ, CODEBOOK DESIGN FROM THE FOUR SEQUENCES: AFTER BIT ALLOCATION.

## A.2 Image sequence coding

The codebook for which the target rate  $R_d = 0.2$  bpp, is used to code the "Salesman" sequence. The maximal time for encoding an image is just about 1.8 s. Figure 18 shows curves obtained when encoding one hundred pictures of the sequence. The PSNR equals in average 36.89 dB, so globally the quality is quite good. The PSNR curve is constant because when the prediction gain becomes less efficient, the quantification gain gets higher (the fall at the beginning of the PSNR curve is due to the stabilization of the closed loop coder). When the rate curves are correlated with gain curves: the peaks appear with the motion in the sequence (because the character moves and handles an object).

The pictures shown in figure 19 are extracted from the sequences. They illustrate how the TSLVQ performs: only the edges of the moving man are finely quantized, while the background and the noise are coarsely quantized.

The curves (figure 18) and the pictures (figure 19) obtained with the "Salesman" sequence are a good representation for this type of results obtained with TSLVQ. Table III displays outcomes when coding the other sequences. These results are too dependent on the coding intrinsic difficulty of the sequence ("Miss America" sequence is simple, "Carphone" sequence not). The adaptive approach aims to overcome this drawback.

## B. Results with adaptive TSLVQ

### B.1 Codebook design

The coder configuration is chosen as before, but the codebook of each sub-band is now designed in two parts. For the stump we use the same training sequence constituted from the four sequences and the same codebook configuration (see figure 17), but the threshold for the bit allocation is very low:  $R_d = 0.1$  bpp (see table II for numerical results). For the branch addition we use the images from the sequence to be coded. As an example, table IV and table V show typical outcomes obtained respectively before and after bit allocation, using the "Salesman" sequence.

The CPU times remains reasonable, and after the bit allocation the rate is closer than the target. The replenishment information is necessary to complete the codebook from the stump, its size is very small.

### B.2 Image sequence coding

We code separately the sequences from their own codebook, namely the training sequence for the branch addition has been constituted only with images from the sequence to be quantized. Table VI shows more regular results: whichever the type of sequence, the final rate is close to the target, and a good quality of reconstruction is obtained. A gap between the target and the final rate remains because the training sequence used for the codebook construction is obtained with an open loop coder, whereas the image sequence coding is achieved next in a closed loop. As a consequence the two vector sources are slightly different. Moreover, we must add that the gain induced by the "dead zone" is subjective, and its size could be settled empirically.

The analysis of results presented as before (encoding time, rate and gain curves, pictures extracted from the sequences) will be the same.

## VII. CONCLUSION

We have proposed a new vector quantizer in the context of video coding applications. The approach aims to unify both efficient coding methods (fast lattice encoding and unbalanced tree-structure codebook) is based on lattice embedding. We have investigated the VQ complete design with: the lattice truncation, the multi-stage procedure of quantization, the unbalanced tree-structured codebook design according to a distortion *v.s.* rate tradeoff, the best lattice determination, the outlying input vector processing and the bit allocation strategy. Taking account of the particular structure of the codebook, the VQ capability for adaptive coding has been explored.

This tree-structured lattice VQ offers some solutions to usual lattice VQ drawbacks with a space partition according to the source distribution, a simple labeling of the lat-

sequence name	number of images	average PSNR	average entropy
"Salesman"	180	36.89 dB	0.195 bpp
"Miss-America"	150	37.32 dB	0.137 bpp
"Carphone"	180	34.25 dB	0.367 bpp

TABLE III

CLASSICAL TSLVQ, CODING OF SEQUENCES WITH THE CODEBOOK DESIGNED FROM THE FOUR SEQUENCES.

subband label	training sequence	CPU time	codeword number	codebook entropy
A	5 images	7.18 s	48	0.954 bpp
B	15 images	5.97 s	104	0.205 bpp
C	15 images	6.11 s	161	0.318 bpp
D	150 images	52.92 s	1348	0.078 bpp

TABLE IV

ADAPTIVE TSLVQ, CODEBOOK DESIGN, BRANCH ADDITION FROM THE "SALESMAN" SEQUENCE: BEFORE BIT ALLOCATION.

subband label	codeword number	codebook entropy	replenishment information
A	20	0.337 bpp	0 bytes
B	104	0.205 bpp	681 bytes
C	86	0.253 bpp	448 bytes
D	92	0.002 bpp	0 bytes

TABLE V

ADAPTIVE TSLVQ, CODEBOOK DESIGN, BRANCH ADDITION FROM THE "SALESMAN" SEQUENCE: AFTER BIT ALLOCATION (THRESHOLD:  $R_d = 0.2$  BPP, FINAL RATE:  $R = 0.199$  BPP).

sequence name	number of images	average PSNR	average entropy
"Salesman"	180	36.66 dB	0.170 bpp
"Miss-America"	150	38.03 dB	0.273 bpp
"Carphone"	180	32.65 dB	0.257 bpp

TABLE VI

ADAPTIVE TSLVQ, CODING OF SEQUENCES WITH THEIR OWN CODEBOOK.

tice points and a fast processing of the outlying input vectors. It is well suited for adaptive video coding because of the "dead zone" in accordance with prediction error coding, the practicable updating of the codebook and the fast image encoding.

## REFERENCES

- [1] R.M. Gray, "Vector quantization," *IEEE ASSP Magazine*, pp. 4–29, April 1984.
- [2] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, 1992.
- [3] T.R. Fisher, "A pyramid vector quantizer," *IEEE Transactions on Information Theory*, vol. 32, no. 4, pp. 568–583, July 1986.
- [4] M. Barlaud, P. Solé, T. Gaidon, M. Antonini, and P. Mathieu, "Pyramidal lattice vector quantization for multiscale image coding," *IEEE Transactions on Image Processing*, vol. 3, no. 4, pp. 367–381, July 1994.
- [5] D.G. Jeong and J.D. Gibson, "Image coding with uniform and piecewise-uniform vector quantizers," *IEEE Transactions on Image Processing*, vol. 4, no. 2, pp. 140–146, February 1995.
- [6] J.H. Conway and Sloane N.J.A., "Voronoi regions of lattices, second moments of polytopes, and quantization," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 211–226, March 1982.
- [7] J.H. Conway and Sloane N.J.A., *Sphere Packings, Lattices and Groups, 2nd edition*, A series of Comprehensive Studies in Mathematics. Springer-Verlag, New York, 1993.
- [8] Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, pp. 84–95, 1980.
- [9] N.M. Nasrabadi and R.A. King, "Image coding using vector quantization : a review," *IEEE Transactions on Communications*, vol. 36, no. 18, August 1988.
- [10] R.M. Gray, *Source coding theory*, Kluwer Academic Publishers, 1990.
- [11] P.A. Chou, T. Lookabaugh, and R.M. Gray, "Entropy-constrained vector quantization," *IEEE Transactions on Acoust. Speech Signal Processing*, vol. 37, no. 1, pp. 31–42, January 1989.
- [12] D.G. Jeong and J.D. Gibson, "Lattice vector quantization for image coding," *Proc. of International Conference on Acoustics, Speech, and Signal Processing ICASSP*, pp. 1743–1746, May 1989.
- [13] P. Zador, "Asymptotic quantization error of continuous signals and their quantization dimension," *IEEE Transactions on Information Theory*, vol. 28, pp. 139–149, March 1982.
- [14] A. Gersho, "Asymptotically optimal block quantization," *IEEE Transactions on Information Theory*, vol. 25, no. 4, pp. 373–380, July 1979.
- [15] T.D. Lookabaugh and R.M. Gray, "High-resolution quantization theory and the vector quantizer advantage," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 1020–1033, September 1989.
- [16] J.H. Conway and Sloane N.J.A., "Fast quantizing and decoding algorithms for lattice quantizers and codes," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 227–232, March 1982.
- [17] F. Kuhlmann and J.A. Bucklew, "Piecewise uniform vector

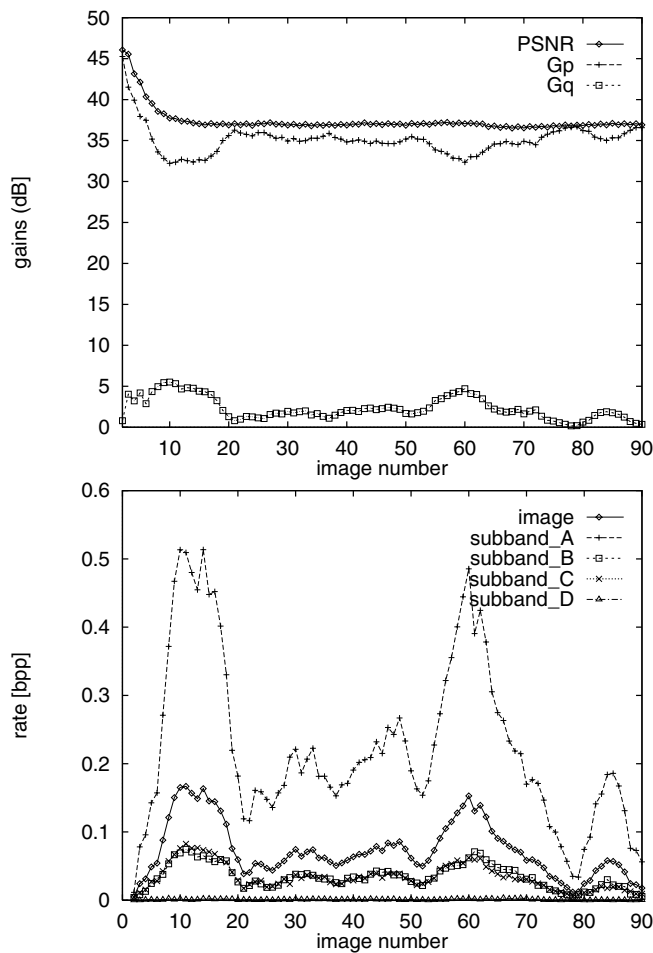


Fig. 18. Coding of the sequence "Salesman": gain and entropy results.

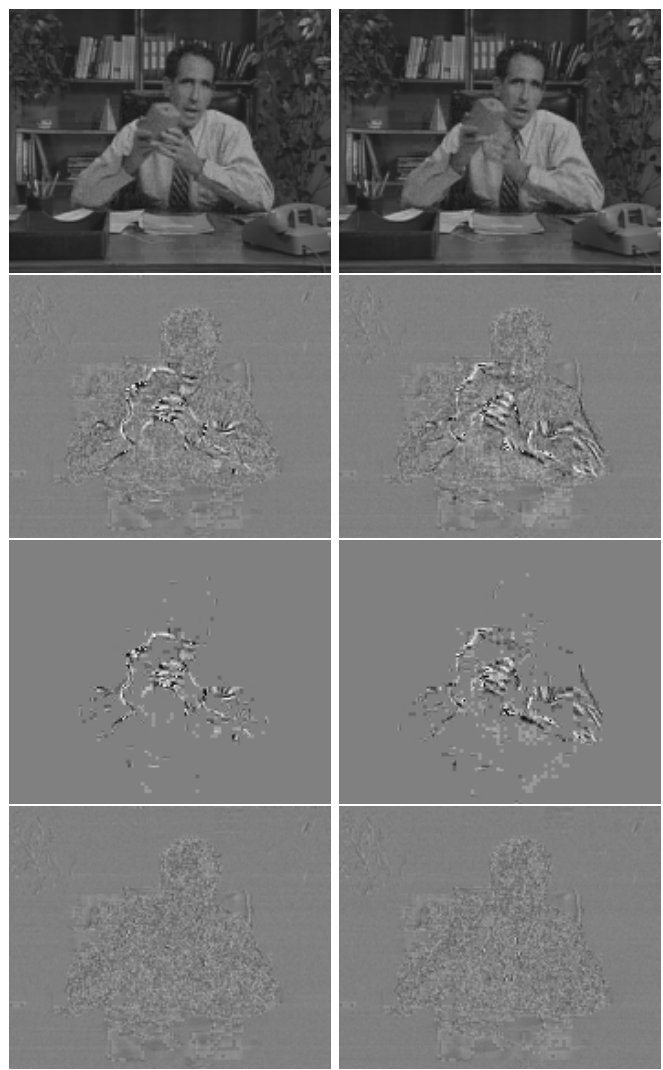


Fig. 19. Images extracted when coding the "Salesman" sequence with from top to bottom: the reconstructed images, the prediction error images (before VQ), the vector quantized prediction error images (after VQ) and the quantization error images. The differential images are centered and scaled three times.

- quantizer," *IEEE Transactions on Information Theory*, vol. 34, no. 5, September 1988.
- [18] D.G. Jeong and J.D. Gibson, "Uniform and piecewise uniform lattice vector quantization for memoryless gaussian and laplacian sources," *IEEE Transactions on Information Theory*, vol. 39, no. 3, May 1993.
- [19] P.F. Swaszek, "Unrestricted multistage vector quantizers," *IEEE Transactions on Information Theory*, vol. 38, no. 3, pp. 1169–1174, May 1992.
- [20] P. Onno, *Bancs de filtres et quantification vectorielle sur réseau : étude conjointe pour la compression d'images*, Ph.D. thesis, Université de Rennes I, March 1996.
- [21] T. Gaidon, *Quantification vectorielle algébrique et ondelettes pour la compression de séquences d'images*, Ph.D. thesis, Université de Nice-Sophia Antipolis, December 1993.
- [22] J.M. Moureaux, *Quantification vectorielle algébrique pour la compression d'images. Application à l'imagerie radar à synthèse d'ouverture (SAR)*, Ph.D. thesis, Université de Nice-Sophia Antipolis, December 1994.
- [23] J.M. Moureaux, M. Antonini, and M. Barlaud, "Lattice vector quantization of image using a product code form and a new labelling method," in *Proc. of Visual Communications and Image Processing VCIP*, September 1994, pp. 422–433.
- [24] J.H. Conway and Sloane N.J.A., "A fast encoding method for lattice codes and quantizers," *IEEE Transactions on Information Theory*, vol. 29, no. 6, pp. 820–824, November 1983.
- [25] C. Lamblin, *Quantification Vectorielle Algébrique Sphérique par le réseau de Barnes-Wall : Application au Codage de la Parole*, Ph.D. thesis, University of Sherbrooke, Quebec, Canada, March 1988.
- [26] M. Antonini, *Transformée en ondelettes et compression numérique des images*, Ph.D. thesis, Université de Nice-Sophia Antipolis, September 1991.
- [27] N.S. Jayant and Noll P., *Digital Coding of Waveforms - Principles and Applications to Speech and Video*, Prentice Hall International Editions, Englewood Cliffs, New Jersey, 1984.
- [28] M. Kunt, A. Ikonopoulou, and M. Kocher, "Second-generation image-coding techniques," *Proc. of the IEEE*, vol. 73, no. 4, pp. 549–574, April 1985.
- [29] A.N. Netravali and B.G. Haskell, *Digital pictures : Representation and Compression*, Plenum Press, New York, 1988.
- [30] J.W. Woods, *Subband image coding*, Kluwer, Boston, 1991.
- [31] G. Tziritas and C. Labit, *Motion analysis for image sequence coding*, vol. 4 of *Advances in image communication*, Elsevier Science Publishers, Londres, July 1994.
- [32] T.R. Fisher, "Geometric source coding and vector quantization," *IEEE Transactions on Information Theory*, vol. 35, no. 1, pp. 137–145, January 1989.
- [33] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 2, April 1992.
- [34] M. Goldberg, P.R. Boucher, and S. Schlien, "Image compression using adaptative vector quantization," *IEEE Transactions on Communications*, vol. 34, no. 2, pp. 180–187, February 1986.
- [35] P. Monet and C. Labit, "Codebook replenishment in classified pruned tree-structured vector quantization of image sequences," in *Proc. of International Conference on Acoustics, Speech, and*

- Signal Processing ICASSP*, 1990, pp. 2285–2288.
- [36] J.E. Fowler, *Adaptive Vector Quantization for the Coding of Nonstationary sources*, Ph.D. thesis, Ohio State University, 1996.
  - [37] J. Pan and T.R. Fisher, “Two-stage vector quantization-lattice vector quantization,” *IEEE Transactions on Information Theory*, vol. 41, no. 1, pp. 155–163, January 1995.
  - [38] C.F. Barnes, S.A. Rizvi, and N.M. Nasser, “Advances in residual vector quantization : A review,” *IEEE Transactions on Image Processing*, vol. 5, no. 2, pp. 226–262, February 1996.
  - [39] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and regression Trees*, The Wadsworth Statistics/Probability Series. Wadsworth, Belmont, California, 1984.
  - [40] P.A. Chou, T. Lookabaugh, and R.M. Gray, “Optimal pruning with applications to tree-structured source coding and modeling,” *IEEE Transactions on Information Theory*, vol. 35, no. 2, pp. 299–314, March 1989.
  - [41] J. Makhoul, S. Roucos, and H. Gish, “Vector quantization in speech coding,” *Proc. of the IEEE*, vol. 73, no. 11, pp. 1551–1588, November 1985.
  - [42] E.A. Riskin and R.M. Gray, “A greedy tree growing algorithm for the design of variable rate vector quantizers,” *IEEE Transactions on Signal Processing*, vol. 39, no. 11, pp. 2500–2507, November 1991.
  - [43] M. Balakrishnan, W.A. Pearlman, and L. Lu, “Variable-rate tree-structured vector quantizers,” *IEEE Transactions on Information Theory*, vol. 41, no. 4, pp. 917–930, July 1995.
  - [44] K.O. Perlmutter, S.M. Perlmutter, R.M. Gray, R.A. Olshen, and K.L. Oehler, “Bayes risk weighted vector quantization with posterior estimation for image compression and classification,” *IEEE Transactions on Image Processing*, vol. 5, no. 2, pp. 347–360, February 1996.
  - [45] Y. Shoham and A. Gersho, “Efficient bit allocation for an arbitrary set of quantizers,” *IEEE Transactions on Acoust. Speech Signal Processing*, vol. 36, no. 9, pp. 1445–1453, September 1988.
  - [46] K. Ramchandran and M. Vetterli, “Best wavelet packet bases in a rate-distorsion sense,” *IEEE Transactions on Image Processing*, vol. 2, no. 2, April 1993.