HIGH DIMENSION LATTICE VECTOR QUANTIZER DESIGN FOR GENERALIZED GAUSSIAN DISTRIBUTIONS

Leonardo Hidd Fonteles and Marc Antonini, Member, IEEE

I3S laboratory, CNRS and University of Nice-Sophia Antipolis
2000, route des Lucioles, B.P. 121, 06903 Sophia-Antipolis Cedex, France
E-mail: fonteles@i3s.unice.fr, am@i3s.unice.fr

ABSTRACT

LVQ is a simple but powerful tool for vector quantization and can be viewed as a vector generalization of uniform scalar quantization. Like VQ, LVQ is able to take into account spatial dependencies between adjacent pixels as well as to take advantage of the n-dimensional space filling gain. However, the design of a lattice vector quantizer is not trivial particularly when one wants to use vectors with high dimensions. Indeed, using high dimensions involves lattice codebooks with a huge population that makes indexing difficult. On the other hand, in the framework of wavelet transform, a bit allocation across the subbands must be done in an optimal way. The use of VQ and the lack of non asymptotical distortion-rate models for this kind of quantizers make this operation difficult. In this work we focus on the problem of efficient indexing and optimal bit allocation and propose efficient solutions.

Index Terms— Image compression, lattice vector quantization, generalized gaussian distribution, bit allocation.

1. INTRODUCTION

Shannon theory implies that the performance of a vector quantizer (VQ[1]) can come arbitrarily close to the theoretical optimal performance if the vector dimension is sufficiently high. Unfortunately, the computational complexity of an unconstrained code increases exponentially with dimension. In addition, the storage requirements can be very large. One solution to overcome this problem of dimensionality is to use some form of constrained VQ such as lattice vector quantization (LVQ) [1]. LVQ is a simple but powerful tool for vector quantization and can be viewed as a vector generalization of uniform scalar quantization. Like VQ, LVQ is able to take into account spatial dependencies between adjacent pixels as well as to take advantage of the n-dimensional space filling gain [2]. Whatever the source distribution is, LVQ will always outperform uniform scalar quantizers. Fast encoding and decoding algorithms making use of simple rounding operations have been proposed by Conway and Sloane [1]. Consequently, the encoding and decoding speed does not depend on the number of codewords within the codebook. Thus, lattice vector quantizers offer the possibility of a substantial reduction in computational and storage complexity over unstructured full-search VQ designed by the GLA algorithm [2]. Its computational simplicity and dictionary robustness make it attractive and widely used in the lossy data compression domain.

The design of an efficient LVQ is not so trivial, particularly if one wants to use vectors with high dimensions and match the source distribution correctly. Indeed, using high dimension vectors involves lattice codebooks with a huge population that makes indexing difficult. On the other hand, in the framework of wavelet transform, a bit allocation across the subbands must be done in an optimal way. The use of VQ and the lack of non asymptotical rate-distortion models for this kind of quantizers make this operation difficult.

In this work we focus on the problem of efficient indexing and optimal bit allocation. The paper is organized as follows. In Section 2 we introduce the principle of LVQ and some backgrounds. Section 3 deals with the problem of indexing. In this section we propose an efficient solution to solve the indexing of huge LVQ codebooks. In Section 4, we propose an efficient approach for the bit allocation. The proposed algorithm works efficiently whatever the quantizer is (scalar quantizer, VQ, LVQ). Finally, Section 5 gives some experimental results and we conclude in Section 6.

2. OVERALL CODING SCHEME

2.1. Principle

The global scheme of the proposed coder lies on the transformation-quantization-compression conventional structure and is presented in the Figure 1.

First of all, a discrete wavelet transform (DWT) is applied on the data which is then split into different coefficient subbands. Then, each of those coefficient subbands is scaled "independently" using a different scaling factor γ. The resulting scaled coefficients are then quantized using a LVQ. The optimal choice of γ for each subband is guaranteed by an efficient resource allocation algorithm (see Section 4 for details). As we will see in Section 2.2, in order to adapt the lattice to the source distribution, each lattice vector is represented by a product code depending on the statistic of the source to be quantized as proposed by Fischer in [3]. Further details on the algorithm are given hereinafter.

2.2. Lattice vector quantization

A lattice Λ in $\mathbb{R}^n$ is composed of all integral combination of a set of linearly independent vectors $\mathbf{a}_i$ (the basis of the lattice) such that:

$$\Lambda = \{\mathbf{x} | \mathbf{x} = u_1\mathbf{a}_1 + u_2\mathbf{a}_2 + ...u_n\mathbf{a}_n\} \quad (1)$$

where the u_i are integers. The partition of the space is hence regular and depends only on the chosen basis vectors $\mathbf{a_i} \in \mathbb{R}^m$ $(m \geq n)$. Note that each set of basis vectors define a different lattice.

Due to the regularity of the lattice, all the lattice vectors with constant l_p norm are lying on concentric shells (or hyper-surfaces). Then, it is possible to encode a given lattice vector using a product code constructed by the concatenation of two indices [3]: the

This work has been supported in part by Alβan Office.

[1] VQ will be used both for vector quantization and vector quantizer.

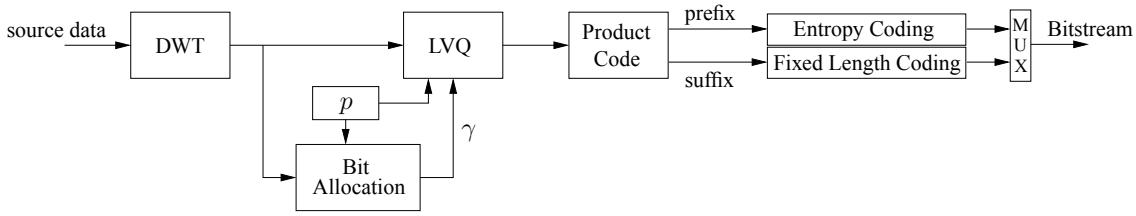

Fig. 1. Global coding scheme. p represents the shape factor of the generalized Gaussian distribution.

first one corresponding to the value of the vector norm r_i (prefix) and the second one corresponding to the position of the vector on the given hyper-surface (suffix). The position of the vector on a given hyper-surface can be found using an enumeration algorithm [1, 3, 4]. Generally, the prefix of the product code is encoded using entropy coding (such as arithmetic coding) and the suffix using a fixed length coding, leading to the bit rate R computed as $R = -\sum_{i=0}^{m-1} p(r_i)[log_2 p(r_i) - \lceil log_2 N(r_i) \rceil]$, where $p(r_i)$ is the probability of a given radius r_i and $N(r_i)$ corresponds to the number of vectors lying on a given hyper-surface obtained by the enumeration algorithm. Such a product code ensures the unicity of decoding.

In the case of sources with generalized Gaussian distribution with a shape parameter less than or equal one (like DWT coefficient sources), the superiority of the cubic $\mathbb{Z}^n$ lattice over D_4, E_8 or leech lattices has been established in [5]. This result demonstrates the great interest in using a combined wavelet transform and a cubic LVQ scheme. In the rest of the paper we will thus focus on the design of a LVQ based on the cubic $\mathbb{Z}^n$ lattice.

3. LATTICE VECTOR INDEXING

3.1. Challenge

VQ is known to give better rate-distortion performances than scalar quantization [2]. Despite the idea behind LVQ be quite simple, it is not the case for the implementation of the product code which remains complex. Indeed, even if the computation of the prefix is trivial, the suffix needs the enumeration and indexing of the lattice vectors lying on a given hyper-surface. The regular indexing method attributes an index to the suffix by taking into account the total number of vectors lying on a given hyper-surface (cardinality). This kind of indexing was introduced by Fischer in [3] for the case of Laplacian distributions and extended to generalized Gaussian distributions by Villasenor in [6]. However, it is well known that the performances of a vector quantizer increase with the vector dimension. In the case of LVQ, increasing the dimension involves an increase of the cardinality of the hyper-surfaces: for some space dimensions and norms, one can have more than a billion lattice vectors lying on a given hyper-surface! Furthermore, this number grows exponentially with the norm. Then, enumeration and indexing of all the vectors can become prohibitive using the algorithms of [3, 6].

A recent approach [7], exploits the *leaders* of a lattice. The leaders of a hyper-surface correspond to few lattice vectors from which all the other lattice vectors lying on the corresponding hyper-surface can be derived by permutations and sign changes of their coordinates. One must note that the number of leaders lying on a given hyper-surface always remains lower than the cardinality of the hyper-surface. The problem of enumeration of [3, 6] is then avoided.

3.2. Indexing based on leaders

The idea behind the method proposed in [7] is to create a suffix based on the construction of a look-up-table and the use of geometrical properties of lattices. This look-up-table contains the index of a few number of vectors, called leaders, from which all the other vectors of the hyper-surface can be assigned taking into account the symmetries of the lattice geometry. These symmetries correspond to two basic operations: changes of the sign of the vector coordinates and permutations of the coordinates. The former one represents the change of hyper-quadrant of the hyper-space in which the vector lies. Instead of indexing directly all the vectors over an hyper-surface, this indexing method assign to each vector a set of three indices: one corresponding to its leader and the two other ones corresponding to its permutation and the sign changes from the leader. For more details in how to compute the permutation and sign indices, see [7].

In order to avoid a heavy computational complexity for indexing the leaders and allowing a direct addressing of the leaders' coordinates, the authors of [7] proposed to construct a look-up-table of dimension of the hyper-space containing all the leaders. This means that for each norm and each lattice dimension, one should store a tensor of high order n, e.g., $n = 12$, $n = 60$ or even more. In addition, this tensor has much more "holes"[2] than the real indices! Indeed, in order to store in a same tensor leaders corresponding to a same dimension and norm, most of the leaders that could be addressed by this table does not exist at all. Furthermore, to construct this table, it is necessary to know or to generate all of the leaders, which remains a complex operation.

For applications where many values of norms and dimensions are necessary, as in the multiresolution data compression context, handling a big amount of huge tensors is quite prohibitive for experimental purposes. In this paper we propose an alternative to this problem while keeping a low-complexity algorithm.

3.3. Proposed indexing in the case of the l_1 norm

3.3.1. Principle

The proposed algorithm classifies all the leader indices in such a way that the indexing is no longer based on a greedy search algorithm or direct addressing, but on low-cost enumeration algorithm which just depends on the *quantity of leaders* instead of on the explicit knowledge of all of them.

A hyper-pyramid of radius r and dimension n is composed by all the vectors $\mathbf{v}$ such that $\|\mathbf{v}\|_1 = r$. As said before, leaders are the elementary vectors of a hyper-surface from which operations of permutations and sign changes lead to all the other vectors lying on this hyper-surface. Indeed, the leaders are vectors with positive

[2] We mean by "hole" a vector which belongs to the table but which is not a leader.

coordinates sorted in increasing (or decreasing) order. Therefore, leaders for l_1 norm are vectors which verify the conditions below:

1. $\sum_{i=1}^{n} v_i = r$;
2. $0 \leqslant v_i \leqslant v_j$, for all $i < j$.

3.3.2. Link with the theory of partitions

In the case of a l_1 norm, one can note that the conditions given in Section 3.3.1 are linked to the *theory of partitions* in number theory [8]. Indeed, in number theory a partition of a positive integer r is a way of writing r as a sum of d positive integers (also called *part*). The number of partitions of r is given by the partition function $p(r)$ such that:

$$\sum_{r=0}^{\infty} p(r) y^r = \prod_{d=1}^{\infty} \left(\frac{1}{1-y^d} \right), \tag{2}$$

which corresponds to the reciprocal of the Euler's function [8]. Further mathematical development lead to representations of the $p(r)$ function that allow faster computation. Interested readers should refer to [8].

For example, for $r = 5$, Equation (2) gives the result $p(5) = 7$. Indeed, in this case, all the possible partitions are: (5), $(1,4)$, $(2,3)$, $(1,1,3)$, $(1,2,2)$, $(1,1,1,2)$ and $(1,1,1,1,1)$. Rewriting these partitions as vectors like $(0,0,0,0,5)$, $(0,0,0,1,4)$, $(0,0,0,2,3)$, $(0,0,1,1,3)$, $(0,0,1,2,2)$, $(0,1,1,1,2)$ and $(1,1,1,1,1)$, we note that they correspond exactly to the leaders of the hyper-pyramid of norm $r = 5$ and dimension $d = 5$, i.e., they are the only vectors which verify the 2 conditions in Section 3.3.1 for the hyper-pyramid of norm $r = 5$ and dimension $d = 5$.

However, we are usually interested in shells of l_1 norm equals to r in a d-dimensional lattice with $r \neq d$. In this case, one can use the function $q(r,d)$ [8] which computes the number of partitions of r with at most d parts (in partition theory it is equivalent to the number of partitions of r with no element greater than d with any number of parts). Then, for a hyper-pyramid of norm $r = 5$ and dimension $d = 3$, we have $q(5,3) = 5$, i.e., five leaders given by: $(0,0,5)$, $(0,1,4)$, $(0,2,3)$, $(1,1,3)$,and $(1,2,2)$.

The function $q(r,d)$ can be computed from the recurrence relation[3] [8]:

$$q(r,d) = q(r,d-1) + q(r-d,d), \tag{3}$$

with $q(r,d) = p(r)$ for $d \geqslant r$, $q(1,d) = 1$ and $q(r,0) = 0$.

3.3.3. Using function $q(r,d)$ to index the leaders

As we will see in the following, Equation (3) not only gives the total number of leaders lying on a given hyper-pyramid but can also be used to provide unique indices for these leaders. To illustrate the principle of the proposed algorithm, let us suppose that the leaders of a given hyper-pyramid have been classified in a lexicographical order as:

Index value	Leader
0	$(0,...,0,0,r_n)$
1	$(0,...,0,1,r_n-1)$
2	$(0,...,0,2,r_n-2)$
3	$(0,...,1,1,r_n-2)$
$\vdots$	$\vdots$

[3] Note that there also exist closed forms for $q(r,d)$ for the first few values of d.

In this way, the index of a leader $\mathcal{L}$ corresponds to the number of leaders that appear before it. For example, the leader $(0,...,1,1,r_n-2)$ should be assigned to index 3.

Consider a leader $\mathcal{L} = (x_1, x_2, ..., x_{n-1}, x_n)$ of dimension n and norm $r_n = \sum_{i=1}^{n} x_i$. Since the leaders are sorted in a lexicographical order, all the leaders with the largest coordinate g_n verifying $x_n + 1 \leqslant g_n \leqslant r_n$ appear before $\mathcal{L}$. The number of leaders with largest coordinate equal to $x_n + t$ ($t \geqslant 1$) and norm r_n can be easily calculated using the function q of Equation (3) and is given by $q(r_n - (x_n + t), n-1)$. Clearly, computing the number of leaders with the largest coordinate equal to $x_n + t$, with norm $r = r_n$ and dimension $d = n$,is equivalent to calculate the number of leaders of norm $r_n - (x_n + t)$ with dimension $n - 1$.

By introducing the function $\overline{q}(r,d,k)$ which counts all he partitions of a number r with at most d parts not greater than k, we can show that the index of a leader can be computed using the following formula:

$$I_{\mathcal{L}} = \sum_{\substack{j=0 \\ \text{while } x_{n-(j+1)} \neq 0}}^{n-2} \sum_{i=x_{n-j}+1}^{\min[x_{n-(j-1)}, r_{n-j}]} \overline{q}(r_{n-j} - i, n-(j+1), i), \tag{4}$$

with $x_{n+1} = +\infty$ and $q(0,d) = \overline{q}(0,d,k) = 1$. Note that, when $r_{n-j} - i$ is less than or equal to i, $\overline{q}(r_{n-j} - i, n-(j+1), i) = q(r_{n-j} - i, n-(j+1))$, because in that case all vectors counted by $q(r,n)$ are leaders.

3.4. Extension to the case of l_p norm

In the case of l_p norms with $0 < p \leqslant 2$, the q-function given by Equation (3) is no longer applicable. Indeed, the l_p norm for a vector $v = (v_1, v_2, ..., v_n)$ involves the computation of $|v_i|^p$ which are not necessarily integers. To circumvent this problem, one can round the values $|v_i|^p$ to their nearest integers with a given precision δ [4] and introduces $\bar{v}_i = round\left(\frac{|v_i|^p}{\delta}\right)$. The l_p norm of a vector is then known with a precision δ.

Let us now define $\overline{q}_\delta^{\alpha}(r,d,k)$ the function which gives the number of partition of $r \in \mathbf{S}_\delta^{p+}$ (where $\mathbf{S}_\delta^{p+}$ is the set composed by all the integers $\bar{v}_i$ with $v_i \in \mathbb{Z}^+$) with largest part equal to k. This function is given by [9]:

$$\overline{q}_\delta^{p}(r,d,k) = \sum_{i=i_{min}}^{k} \overline{q}_\delta^{p}\left(r - \left\lceil \frac{i^p}{\delta} \right\rceil, d-1, i\right) \tag{5}$$

with $\overline{q}_\delta^{p}(0,1,1) = 1$. The value i_{min} is the lowest value of $i \in \mathbb{Z}$ verifying $\left\lceil \frac{i^p}{\delta} \right\rceil \geqslant \left\lceil \frac{r}{d} \right\rceil$. Then, replacing $\overline{q}(r,d,k)$ in Equation (4) by $\overline{q}_\delta^{p}(r,d,k)$ permits to assign an index to a leader $\mathcal{L}$ with l_p norm equal to r with precision δ.

3.5. Decoding a leader index

Let us suppose that the index $I_{\mathcal{L}}$ of the leader to be decoded as well as its l_p norm r are transmitted to the decoder. Furthermore, let us assume that the vector size d, the shape factor p as well as the precision δ are known at the decoder side. Then, since the first leader of size d and norm r is always given by $(0,0,...,0,r)$, it is possible to easily retrieve the coordinates of the leader corresponding to index $I_{\mathcal{L}}$ using a counting algorithm as proposed for indexing in

[4] A similar approach was used in [6]

Section 3.3. Indeed, the decoding algorithm must count the number of leaders with largest coordinate x_d varying from r down to $\lceil r/d \rceil$ while the index $I_{\mathcal{L}}$ is not reached.

4. MODEL-BASED LVQ BIT ALLOCATION

4.1. Optimal bit allocation

To solve the problem of bit allocation, we propose to use a Lagrangian approach and to introduce the following Lagrangian functional $J(\mathbf{R}, \lambda)$:

$$J(\mathbf{R}, \lambda) = \sum_{i=1}^{M} w_i D_i(R_i) - \lambda(\sum_{i=1}^{M} a_i R_i - R_{\max}) \qquad (6)$$

where D_i and R_i are respectively the distortion and the rate in each subband i, and R_{max} the rate budget. By imposing the zero-gradient condition, the resulting optimal rate allocation vector $\mathbf{R}^* = \{R_i^*\}_{i=1}^M$ must verify the following set of equations:

$$\frac{w_i}{a_i} \frac{\partial D_i}{\partial R_i}(R_i^*) = \lambda \qquad \forall i \in \{1, \ldots, M\} \qquad (7)$$

where λ is the Lagrange multiplier. We can read (7) this way: the optimal rates correspond to points having the same slope on the "weighted" curves $(R_i, \frac{w_i}{a_i} D_i)$. Note that $\lambda < 0$ since the RD curve are strictly decreasing. We proposed in [10] an efficient algorithm to compute the values of R_i.

4.2. Model for $D(\mathbf{R})$ curves

The method proposed in Section 4.1 not only requires the knowledge of the RD curve of each subband, but also supposes that these curves are differentiable, convex and accurate enough. The estimation of the RD curves is thus a crucial step. To that purpose, we evaluate experimentally the quantizer (R,D) points located on the convex hull of the RD function and distributed between an ad hoc range of bitrates[5]. Then, these real RD curves are modeled using smoothing B-Splines providing a continuous, convex and differentiable analytical model for each RD function of each subband [10]. This approach can be seen as both data-driven and model-based.

5. EXPERIMENTAL RESULTS

The performances of the proposed method are evaluated for image coding in the framework of discrete wavelet transform (DWT). The LVQ is designed for a cubic $\mathbb{Z}^n$ lattice for a generalized Gaussian distribution (l_p norm). In the case of l_p norm, the shape parameter p is estimated according to the DWT sub-band. The coding of the lattice vectors uses a product-code with an arithmetic coding for the vector norms. The leader indices are coded with a variable length coding (with length depending on the population of each orbit or class[6]), while fixed length coding is used for permutation and sign indices. The size of the vectors across the different DWT sub-bands are chosen such that a good bit allocation is obtained. In the following example, the vector sizes are chosen in a range from 8 to 128.

At low bit rates the proposed method gives promising results when compared to standard codec such as JPEG-2000 with contextual arithmetic coding.

[5]The range depends on the bandwith of the subband, i.e., for high frequency subbands small bitrates will be allocated while for low frequency subband high bitrates will be allocated.

[6]The set of vectors having the same leader $\mathcal{L}$ is the *orbit* or *class* of $\mathcal{L}$.

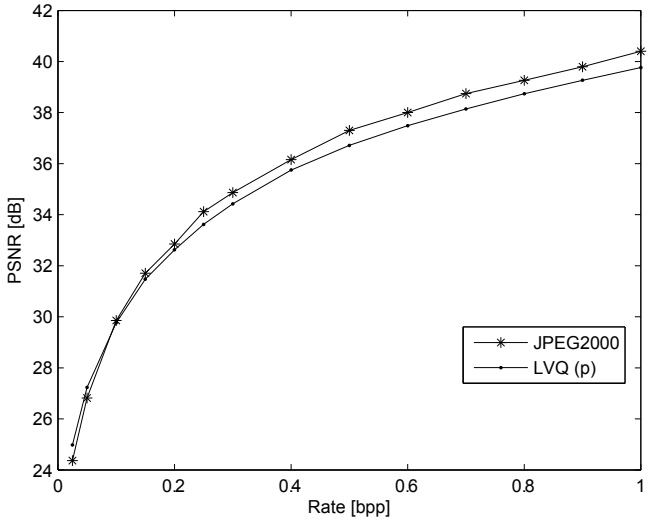

Fig. 2. PSNR vs Rate for the image LENA 512×512

6. CONCLUSION

In this paper we have proposed an efficient indexing solution for a $\mathbb{Z}^n$ lattice and for l_p norms. The l_p norms with $0 \leqslant p \leqslant 2$ are adapted to sources with generalized Gaussian distributions as DWT sub-bands. In the proposed scheme, indexing a lattice vector is reduced to indexing its corresponding leader, a permutation and a sign change. The number of leaders being smaller than the cardinality of a hyper-surface, indexing a leader can be done even for huge vector sizes and codebook sizes, and doesn't suffer from any computer precision requirement or memory utilization. Furthermore, this analytical indexing is performed without the need of any look-up-table (as was proposed in [7]). Experimental results are promising.

7. REFERENCES

[1] J. Conway and N. Sloane, *Sphere packings, lattices and groups*. Springer Verlag, 1988.

[2] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Press, 1992.

[3] T. R. Fischer, "A pyramid vector quantizer," *IEEE Trans. Inform. Theory*, vol. 32, 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 Trans. Image Processing*, vol. 3, no. 4, pp. 367–381, july 1994.

[5] Z. Gao, F. Chen, B. Belzer, and J. Villasenor, "A comparison of the $\mathbb{Z}$, E8, and leech lattices for quantization of low-shape-parameter generalized gaussian sources," *IEEE Signal Processing Letters*, vol. 2, pp. 197–199, October 1995.

[6] Z. G. F. Chen and J. Villasenor, "Lattice vector quantization of generalized gaussian sources," *IEEE Transactions on Information Theory*, vol. 43, pp. 92–103, january 1997.

[7] J. Moureaux, P. Loyer, and M. Antonini, "Low complexity indexing method for $\mathbb{Z}^n$ and D_n lattice quantizers," *IEEE Transactions on Communications*, vol. 46, no. 12, pp. 1602–1609, december 1998.

[8] G. E. Andrews, "The theory of partitions," Cambridge University Press; Reprint edition (July 28, 1998), 1998.

[9] L. H. Fonteles and M. Antonini, "Indexing $\mathbb{Z}^n$ lattice vectors for generalized gaussian distributions," in *Proc. of IEEE ISIT*, June 2007.

[10] ——, "Lattice vector quantization for normal mesh geometry coding," *Proc. of IEEE ICASSP*, May 2006.