A Simple Algorithm for Ordering and Compression of Vector Codebooks

Maciej Bartkowiak and Adam Łuczak

Poznań University of Technology, Institute of Electronics and Telecommunications
ul. Piotrowo 3A, 60-965 Poznań, Poland, tel. (+4861) 6652171, fax (+4861) 6652572
{mbartkow, aluczak}@et.put.poznan.pl

Abstract. The problem of storage or transmission of codevectors is an essential issue in vector quantization with custom codebook. The proposed technique for compression of codebooks relies on structuring and ordering properties of a binary split algorithm used for codebook design. A simple algorithm is presented for automatic ordering of the codebook entries in order to group similar codevectors. This similarity is exploited in efficient compression of the codebook content by the means of lossless differential coding and lossy DCT-based coding. Experimental results of two compression experiments are reported and show that a small compression gain can be achieved in this way.

1 Introduction

Various applications of vector quantization in image data compression are studied for years. Out of two scenarios which are using an universal codebook and using a custom codebook, the latter offers significantly better quality of reconstructed images at the cost of additional storage and/or transmission of the codebook, which is prohibitive in achieving high compression efficiency. Therefore some coding algorithms are being investigated for more efficient representation of the codebook content.

Hereafter, we refer to the codebook $\mathbf{X} = \{\underline{X}_1, \underline{X}_2, \ldots \underline{X}_N\}$ as a set of vectors $\underline{X}_i = [x_1^i, x_2^i, \ldots x_K^i]^T$ calculated from image samples, subband/wavelet samples, etc. However following discussion is not limited to any particular data representation, a simple direct vector quantization of the blocks of pixels is used as a benchmark for experimental simulations.

2 Tree-structured codebooks

Among several approaches to codebook design, tree-structured techniques which result in tree-structured codebooks exhibit important advantages over non-structured ones, at the cost of slightly reduced performance of both the codebook design and the

actual vector quantization stages. First, designing an optimal codebook in highly-dimensional space using an optimization method is very difficult and computationally intractable, while hierarchical methods (such as these employed in tree-structured techniques) offer significant savings in computational complexity. Second, tree-structured codebook allows for a simplified the actual vector quantization stage, whereby, instead of full search, the input vector is hierarchically compared to the successive nodes of the tree.

Codebook design based on tree-structured algorithm is usually an iterative procedure of hierarchical partitioning of the multidimensional data space into disjoint regions. Starting from just one codevector representing the whole data set, the codebook grows in each step. One cluster of vectors in a particular region is split in each iteration, and a respective codevector is replaced by two or more vectors representing the new regions just formed, usually centroids of data clusters. A very popular variant of this approach is a binary split (BS) technique, wherein the region under consideration is always split into two subregions and two new clusters are formed (cf fig. 1), which is associated with codebook growth by 1 codevector:

$$\mathbf{X}^{(n+1)} = \mathbf{X}^{(n)} \cup \{\underline{X}_{i'}, \underline{X}_{i''}\} - \{\underline{X}_i\} \qquad (1)$$

where $\mathbf{X}^{(n)}$ denotes the codebook in *n*-th stage of the binary split algorithm,
$\underline{X}_i$ denotes the codevector being replaced by two new codevectors, $\underline{X}_{i'}$ and $\underline{X}_{i''}$.

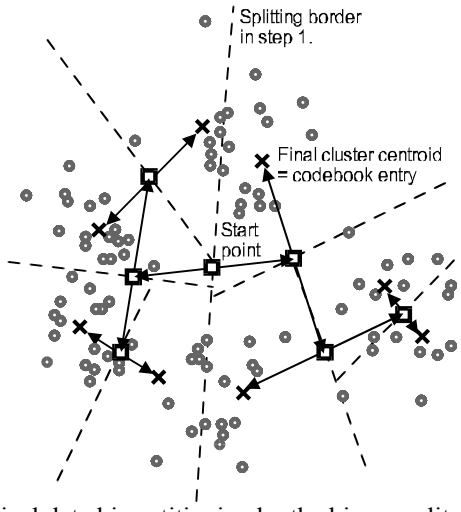

Fig. 1. Hierarchical data bi-partitioning by the binary split algorithm during consecutive steps of the BS algorithm (2D case)

Various rules of region splitting may be applied to the BS algorithm. The most typical approach is to split the region along a hyperplane passing through its centroid in some strictly defined direction related to the data distribution within the region. One of the simplest approaches, proposed by Linde, Buzo and Gray [1] relies on the strengths of their generalized Lloyd algorithm (GLA) known as LBG algorithm (which is similar

to the K-means algorithm of MacQueen, [2]) for K=2. It consists of simple iterative LBG-based refinements of randomly initiated split.

An important near-optimal solution (cf. e.g. [3,4,5]) is derived from the principal component analysis (PCA). The cluster in question is being split along the direction of maximum data variation around its centroids which is determined by the dominant eigenvector of the data covariance matrix within the cluster.

3 Codebook compression: existing approaches and the new proposed technique

Codebook compression is often considered as a method for effective increasing the efficiency of data coding based on vector quantization [6]. In general, two approaches are practiced: lossless and weakly-lossy coding. Straightforward Huffman coding of the component values of codevectors brings marginal (if any) compression gain (cf Table 1) due to their distribution being almost uniform.

Experiments show, that lossy compression utilizing quantization often distorts the codebook in way that makes it no longer optimal for quantizing the image it has been primarily designed for, and thus the whole process loses the advantage of using a custom codebook. Our proposal assumes the codebook is coded without any loss or with only marginal distortion so that application of custom codebook is still justified.

Table 1. Compression gain (over PCM representation) offered by direct application of Huffman coding to the values of codebook vectors

Codebook size	LENA		BOATS	
	Huffman code size	Compression gain [%]	Huffman code size	Compression gain [%]
4x4x128	18938	0%	18855	0%
4x4x256	34401	0%	34310	0%
4x4x512	65243	0.4%	65042	0.7%
4x4x1024	127097	3%	126964	3.1%
4x4x2048	249991	4.6%	249249	4.9%
4x4x4096	495895	5.4%	492514	6%

Experiments show, that certain pairs of codevectors usually exhibit mutual similarity. This similarity may be exploited in various ways through application of data compression techniques, e.g. predictive coding or transform coding, applied across vectors, i.e. along a sequence of ordered vectors, as opposed to intra-vector coding. The efficiency of such compression strongly depends on the correlation between consecutive members of the sequence, therefore the effort is directed toward defining an ordering rule,

$$\Omega : \{\underline{X}_i\} \rightarrow \{S_i\}, \quad 1 < S_i < N \tag{2}$$

that maximizes this correlation. Such rule is in fact the one that minimizes the total length of a chain of ordered codevectors,

$$\Omega_{opt} = \arg\min_{\Omega} \sum_{i=1}^{N-1} \left\| \underline{X}_{S_i} - \underline{X}_{S_{i+1}} \right\| \tag{3}$$

Such order may be determined through exhaustive search, which however would be unreasonable considering the little gain in compression ratio achieved.

4 Automatic codebook ordering

The proposed simple technique relies on structuring and ordering properties of a binary tree. The algorithm is based on a observation that, for reasonably sized tree-structured codebooks, in most cases the aforementioned similarity is mostly observed between pairs of codevectors belonging to the same part of the tree. Therefore the ordering is performed automatically during the codebook design process and is defined by the two inductive rules:

1. Before each step of the binary split algorithm, the codebook in its current stage of growth is assumed to be already ordered (which is trivial for the first and the second step, when codebook consists only of one or two codevectors, respectively),

$$\mathbf{X}^{(n)} = \left\{ \underline{X}_{S_1}, \underline{X}_{S_2}, \ldots \underline{X}_{S_i}, \ldots \underline{X}_{S_N} \right\} \tag{4}$$

2. Each replacement of the codevector $\underline{X}_{S_i}$ with two new codevectors, $\underline{X}_{S_i'}$ and $\underline{X}_{S_i''}$, is associated with inserting two new members into the existing sequence, in the place of the old member. Out of two possibilities, the one is chosen that results in lower the total chain length (cf. Fig. 2):

$$\left\| \underline{X}_{S_{i-1}} - \underline{X}_{S_i'} \right\| + \left\| \underline{X}_{S_i''} - \underline{X}_{S_{i+1}} \right\| < \left\| \underline{X}_{S_{i-1}} - \underline{X}_{S_i''} \right\| + \left\| \underline{X}_{S_i'} - \underline{X}_{S_{i+1}} \right\| \quad \Rightarrow \tag{5a}$$
$$\Rightarrow \quad \mathbf{X}^{(n+1)} = \left\{ \underline{X}_{S_1}, \ldots \underline{X}_{S_i'}, \underline{X}_{S_i''}, \ldots \underline{X}_{S_N} \right\}$$

or

$$\left\| \underline{X}_{S_{i-1}} - \underline{X}_{S_i'} \right\| + \left\| \underline{X}_{S_i''} - \underline{X}_{S_{i+1}} \right\| > \left\| \underline{X}_{S_{i-1}} - \underline{X}_{S_i''} \right\| + \left\| \underline{X}_{S_i'} - \underline{X}_{S_{i+1}} \right\| \quad \Rightarrow \tag{5b}$$
$$\Rightarrow \quad \mathbf{X}^{(n+1)} = \left\{ \underline{X}_{S_1}, \ldots \underline{X}_{S_i''}, \underline{X}_{S_i'}, \ldots \underline{X}_{S_N} \right\}.$$

Thanks to hierarchical partitioning scheme, major data clusters are separated in first order into different branches of the binary tree, so are separated within the sequence the codevectors representing them. As the smallest clusters are divided last, the corresponding centroids codevectors most likely result being placed next to each other in the sequence.

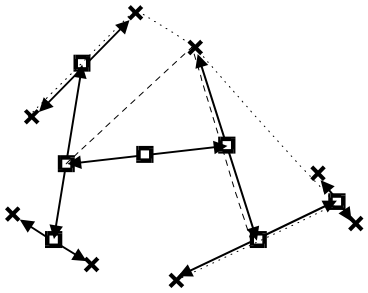

Fig. 2. Example ordering of the tree nodes from Fig. 1 after second step (dashed line) and fourth step (dotted line) of the binary split algorithm

5 Application to codebook compression

The proposed coding techniques- lossless differential coding and lossy transform coding both make use of the increased correlation between consecutive codevectors that results from the proposed ordering. For the purpose of coding, *K* data vectors of length *N* are formed by scanning the corresponding values x_j^i across all codevectors:

$$\underline{U}_j = \left[x_j^1, x_j^2, x_j^3, \ldots x_j^N\right], \quad j = 1 \ldots K \,. \tag{6}$$

In case of predictive differential coding, difference vectors are calculated, for each pair of neighboring codevectors in the ordered sequence. For the first vector, its difference with the average value of its coordinates is calculated.

$$\begin{aligned} \underline{D}_i &= \underline{X}_i - \underline{X}_{i-1}\,, \quad i = 2 \ldots N \\ \underline{D}_1 &= \underline{X}_1 - \tfrac{1}{K}\sum_{j=1}^{K} x_1^j \,. \end{aligned} \tag{7}$$

Subsequently, the set of differential values is encoded using Huffman code with custom dictionary.

Lossy encoding of the codebook data involves calculating discrete cosine transform of the vectors $\underline{U}_j$, quantization and Huffman coding:

$$\begin{aligned} \left[F_k\right]_j^T &= \mathbf{C}\,\underline{U}_j{}^T \\ Q_k &= \begin{cases} F_k & k = 0 \\ \left\lfloor \dfrac{F_k}{q} \right\rfloor & k > 0 \end{cases} \end{aligned} \tag{8}$$

where **C** denotes the DCT matrix, Q_k denotes the quantized DCT coefficient, and q denotes the quantization factor. Due to weak quantization, zero-valued coefficients do not occur as frequently as they do in case of typical image transform coding. Therefore, LRL coding does not offer significant gain in compression ratio.

6 Experimental results

Three series of experiments have been performed in order to estimate the influence of codebook ordering on the compression gain. Codebooks consisting of 128, 256, 512, 1024, 2048 and 4096 vectors were designed with a PCA-based binary split algorithm using samples of test images partitioned onto 4x4 blocks. The codebooks were ordered using the proposed algorithm and further encoded using both lossless differential coding and lossy DCT-based coding.

Table 2. Compression gain resulting from application of lossless differential coding of codevectors after codebook ordering using the proposed algorithm

Codebook size	LENA		BOATS	
	Huffman code size	Compression gain [%]	Huffman code size	Compression gain [%]
4x4x128	14558	11%	15501	5.4%
4x4x256	27267	16.7%	29420	10%
4x4x512	52356	20%	55677	15%
4x4x1024	100612	23%	109329	16.6%
4x4x2048	199400	23.9%	213939	18.4%
4x4x4096	384797	26.6%	414979	20.8%

As shown in table 2, application of differential coding to codebook compression brings small improvement in terms of positive compression gain (few percent to over 25 percent of the number of bits required to represent the codebook) which is achieved without any decrease of the quality of reconstructed codebook (and hence without any loss in image quality). This compression gain grows especially for larger codebooks.

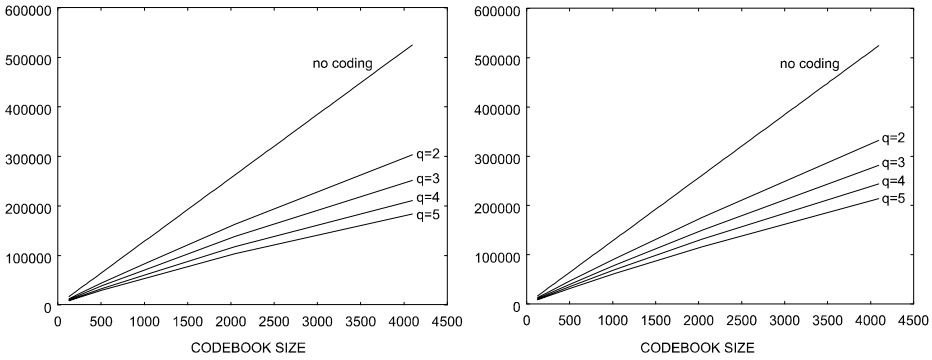

Fig. 3. Experimental results for lossy DCT-based coding of codebooks for test images LENA (left plot) and BOATS (right plot): total number of bits required to store the codebook versus the codebook size (number of codevectors) for various quantization coefficients

Compression gain offered by lossy DCT-based coding scheme is much larger (over 50% and more). The interesting conclusion is that the gain only very slightly grows with the codebook size (cf fig. 3).

The last experiment used codebooks reconstructed after lossy DCT-based coding to vector quantize the original images. Comparison of overall performance of the coding scenarios with and without codebook compression shows that the differences are small. Only for very small values of quantization factor, q=2 , one can observe a consistent improvement in coding efficiency. In case of stronger quantization of the DCT coefficients, the distortions introduced to the codebook make it unfortunatelly no longer optimal and decrease the quality of reconstructed images.

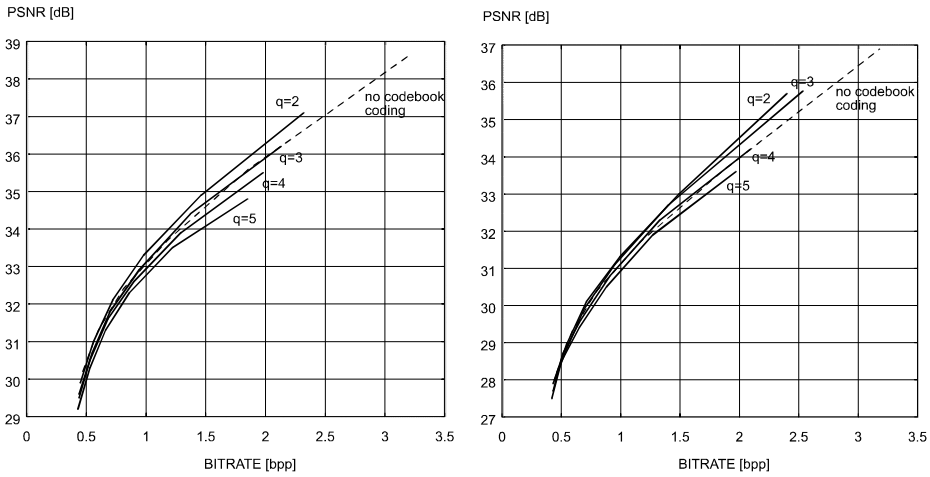

Fig. 4. Comparison of overall vector quantization performance without (dashed line) and with (solid lines) lossy codebook compression for test images LENA (left plot) and BOATS (right plot)

7 Conclusions

The paper proposes a simple algorithm to order the codevectors during tree-structured codebook design, which results in increased the similarity between neighboring vectors. This similarity is exploited by differential and DCT-based coding applied across vectors. Experiments show that small increase of the coding efficiency can be achieved with lossless or weakly lossy coding of codevectors. Efficient compression of codebooks is very difficult, however. We reckon that codebooks obtained form PCA-based binary splitting algorithm are close to optimal and therefore heavily decorrelated.

References

1. Linde Y., Buzo A., Gray R. M., An Algorithm for Vector Quantizer Design, IEEE Trans. on Communications, (1980), 84-89
2. MacQueen J. B., Some Methods for Classification and Analysis of Multivariate Observations, Proc. 5th Berkeley Symp. Math. Statistics and Probability (1967), 281-297

3. Morgan J. N., Sonquist J. A., Problem in the Analysis of Survey Data, and a Proposal, J. Amer, Static. Assoc., **58**, (1963), 415-434
4. Orchard M., Bouman C., Color Quantization of Images, IEEE Trans. on Sig. Proc., **39**, 12 (1991), 2677-2690
5. Wu X., Zhang K., A Better Tree-Structured Vector Quantizer, Proc. IEEE Data Compression Conf, IEEE Computer Society Press, LA, (1991), 392-401
6. Dionysian R., Ercegovac M., Vector quantization with compressed codebooks, Signal Processing: Image Communication 9 (1996), 79-88