LEARNING DICTIONARIES FOR MATCHING PURSUITS BASED VIDEO CODERS

Philippe Schmid-Saugeon and Avideh Zakhor

Berkeley, CA

ABSTRACT

In this paper, we present a learning scheme for designing dictionaries of two-dimensional functions for *matching pursuits* (MP) based video coding. The motivation is to improve the performance of such codecs by adapting the structure of the dictionary functions to specific bit-rates or types of sequences. The scheme we propose is based on *vector quantization* (VQ), and uses an inner-product based distortion measure. The different processing steps, consisting of data extraction from the motion compensated error frames, training, pruning, and testing, are presented in detail. We find that for high bit-rate QCIF sequences we can achieve improvements of up to 0.66 dB.

1. INTRODUCTION

Matching pursuits (MP) based video codecs have been subject of much attention in the literature in recent years [1, 2]. These codecs are identical to hybrid motion-compensated block-based *discrete cosine transform* (DCT) codecs, except that they use an algorithm called MP [3] to encode the motion compensated error, thus avoiding noticeable distortions and blocking artifacts. Outstanding performance has been obtained with this compression scheme at very low bit-rates [1].

MP expands a signal over an *over-complete* dictionary of *normalized* functions in an iterative fashion. The matching is based on the inner product between the signal and a given dictionary function; the updated signal, called *residual*, is computed by subtracting the best matching function, called hereafter *atom*. The residual is initially set to the motion compensated error in the case of video coding. The different definitions and equations used in MP can be found in [3] and will not be repeated here.

In most of the MP based video codecs reported in the literature, a set of separable *Gabor* functions is used as dictionary. This leads to a fast implementation of the MP algorithm but has a number of drawbacks: there is no orientation information, and Gabor atoms have a tendency to introduce small oscillations in the reconstructed signal, especially when the number of atoms is small, i.e. at very low bit-rates. Most of the reported investigations deal with the optimization of the system using Gabor dictionaries [4, 5]. Chou et al. [6] use gain-shape vector quantization to learn new dictionaries. However, their approach does not exploit characteristics of the MP algorithm. In our investigation we choose to learn a dictionary using motion compensated residuals obtained from a set of training sequences, and to adapt the learning scheme to the characteristics of MP.

This paper is organized as follows: Section 2 describes our methodology and the learning scheme we have developed. Section 3 describes our simulation strategy, and Section 4 shows results. Finally, conclusions are drawn in Section 5.

2. LEARNING SCHEME

Our learning scheme is based on vector quantization (VQ) [7]. It is an iterative algorithm that learns a given number of vectors, called hereafter *code-vectors*, from a set of input vectors, called hereafter *patterns*, according to a pre-defined distortion measure. Each iteration has two processing steps:

1. Partition the set of patterns.
2. Update the code-vectors in order to minimize the total distortion in each partition.

The algorithm ends when a predefined stopping criterion, such as a maximum allowed overall distortion, is met.

MP uses the inner product to match the different dictionary functions to the residuals and to select the different atoms used to encode the original signal. We have therefore chosen to use an inner product based distortion measure in our VQ scheme, since this metric will later define how well a learned dictionary function matches a residual. Let $\mathbb{S} \subset \mathbb{R}^k$ be a set of M normalized training patterns of dimension k, $\mathbb{X} = \{1, \ldots, N\}$ the set of all code-vector indices, and n the iteration number. The energy ω_i of the i^{th} pattern is computed before normalization for later use during the code-vector updating step. We define the following distortion measure between a normalized pattern $\mathbf{x}_i \in \mathbb{S}$ and the j^{th} normalized code-vector $\hat{\mathbf{x}}_{j,n}$:

$$d_{\langle\cdot,\cdot\rangle}(\mathbf{x}_i, \hat{\mathbf{x}}_{j,n}) = 1 - |\langle\mathbf{x}_i, \hat{\mathbf{x}}_{j,n}\rangle|\,, \tag{1}$$

where $< \cdot, \cdot >$ is the inner product. The distortion is equal to 1 when $\mathbf{x}_i$ and $\hat{\mathbf{x}}_{j,n}$ are orthogonal and to zero when they are identical. A partition $\mathbb{S}_{j,n}$ is a set of patterns having minimum distortion with respect to a given code-vector $\hat{\mathbf{x}}_{j,n}$:

$$\mathbb{S}_{j,n} = \left\{ \mathbf{x}_i \in \mathbb{S} \mid d_{\langle \cdot,\cdot \rangle}(\mathbf{x}_i, \hat{\mathbf{x}}_{j,n}) \leq d_{\langle \cdot,\cdot \rangle}(\mathbf{x}_i, \hat{\mathbf{x}}_{l,n}) \ , \ \forall l \in \mathbb{X} \right\} \ , \tag{2}$$

and

$$\mathbb{S} = \bigcup_{j \in \mathbb{X}} \mathbb{S}_{j,n} \,, \tag{3}$$

$$\mathbb{S}_{j,n} \cap \mathbb{S}_{l,n} = \emptyset \,, \tag{4}$$

$\forall j \neq l$ and with $j, l \in \mathbb{X}$.

The updated code-vector $\hat{\mathbf{x}}_{j,n+1} \in \mathbb{R}^k$ is obtained by minimizing the total distortion $\delta_{j,n}$ in $\mathbb{S}_{j,n}$:

$$\begin{aligned} \delta_{j,n} &\equiv \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} d_{\langle \cdot,\cdot \rangle}(\mathbf{x}_i, \hat{\mathbf{x}}_{j,n+1}) && (5) \\ &\leq \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} d_{\langle \cdot,\cdot \rangle}(\mathbf{x}_i, \mathbf{x}) \quad , \forall \mathbf{x} \in \mathbb{R}^k \ . && (6) \end{aligned}$$

Since both $\mathbf{x}_i$ and $\hat{\mathbf{x}}_{j,n}$ are normalized, the following L_2-norm distortion measure can be used instead of Eq. 1:

$$\begin{aligned} d_{L_2}(\mathbf{x}_i, \hat{\mathbf{x}}_{j,n}) &= \|\hat{\mathbf{x}}_{j,n} - \mathbf{x}_i\|^2 \\ &= (\hat{\mathbf{x}}_{j,n} - \mathbf{x}_i) \cdot (\hat{\mathbf{x}}_{j,n} - \mathbf{x}_i)^T \\ &= 2 - 2\hat{\mathbf{x}}_{j,n} \cdot \mathbf{x}_i^T \\ &= 2(1 - \langle \mathbf{x}_i, \hat{\mathbf{x}}_{j,n} \rangle) \ , \end{aligned} \tag{7}$$

provided all inner products are positive. To achieve this, we let each pattern have two equivalent versions: the original and its negative, i.e. $\mathbf{x}_i$ and $-\mathbf{x}_i$. This is possible because Eq. 1 uses the absolute value of the inner product. We then define $\mathbb{S}_{j,n}^{(+)}$ and $\mathbb{S}_{j,n}^{(-)}$ as the sets of patterns in $\mathbb{S}_{j,n}$ having respectively positive and negative inner product with $\hat{\mathbf{x}}_{j,n}$:

$$\begin{aligned} \mathbb{S}_{j,n}^{(+)} \cup \mathbb{S}_{j,n}^{(-)} &= \mathbb{S}_{j,n} \,, && (8) \\ \mathbb{S}_{j,n}^{(+)} \cap \mathbb{S}_{j,n}^{(-)} &= \emptyset \,. && (9) \end{aligned}$$

Once both subsets are computed, we can use Eq. 7 instead of Eq. 1 by taking the negative value of the inner product for each pattern in $\mathbb{S}_{j,n}^{(-)}$. We have shown using Lagrange multipliers that the minimization of Eq. 5 with the distortion measure defined in Eq. 7 leads to the following weighted average update equation:

$$\hat{\mathbf{x}}_{j,n+1} = \frac{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i \mathbf{x}_i - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i} \ . \tag{10}$$

More weight is given to high energy patterns in Eq. 10, since it is essential to first encode high energy structures present in the motion compensated error. The code-vectors are normalized after being updated.

The VQ algorithm described so far usually converges to a local minimum [7]. In our scheme, we propose to put a constraint on the partition size according to a monotonically decreasing function of the iteration number. Partitions smaller than the value given by this function are eliminated. In order to keep the same number of centroids, a randomly selected partition is split into two, with larger partitions being more likely to be selected than smaller ones. The following exponential threshold function is used in our simulations:

$$\Omega_{\text{thresh}} = \frac{\Omega}{N} \exp\left\{ -\frac{M}{M_0} \right\} \ , \tag{11}$$

where M is the iteration number, M_0 is a constant scalar that controls the time necessary to converge to the final solution, N is the number of code-vectors, and $\Omega = \sum_{i=1}^{n} \omega_i$ is the weighted size of the pattern space. In our simulations we set $M_0 = 20$, and Ω_{thresh} is only used every fourth iteration in order to allow the system to stabilize in the neighborhood of a local minimum. Otherwise it is set to zero. While this approach is of low complexity, it has shown to be robust, and to lead to near-optimal results.

The extraction of training patterns from the motion residuals is an important issue. The entire residual cannot be learned by the VQ scheme since the high energy content is sparsely distributed. Only regions in the residual where one or several dictionary functions are matched are taken into account. The patterns used to learn new functions are extracted from a set of training sequences encoded with an initial dictionary, in our case a dictionary of Gabor functions. Each time a dictionary function is matched to the residual, the underlying pattern is extracted. We use a square window with a fixed size, centered on the matched function. Using this approach, only high energy regions of the residual are used for the training. The initial dictionary, called *h30* [8], contains 400 separable Gabor functions and 72 non-separable Gabor functions. The number of functions learned in our simulations is therefore always 472.

Since patterns might be extracted from a region where a dictionary function has been previously matched, their content might be influenced by the dictionary itself. In addition to that, residuals in successive frames depend on the dictionary used to encode the previous frames, and so do the extracted patterns. We are therefore facing a "chicken and egg" problem, and in order to have a rigorous learning scheme we need to repeat the *pattern extraction-VQ* cycle several times, using the successively learned dictionaries to extract a new set of patterns. This is a very time consuming process, and we have experienced that running only one cycle already takes a long time. We have run several cycles in some of our simulations and have found no noticeable difference in performance between successive cycles. As such, we have chosen to run only one cycle, especially for the most time consuming simulations. However, future studies are needed to better understand the effect of successive cycles.

Finally, note that once a new dictionary has been learned, the training sequences are encoded with this new dictionary in order to produce usage statistics. These statistics are then used to compute the Huffman codes necessary to encode the atom parameters for the test sequences.

3. SIMULATIONS

Three dictionaries are learned, each one having a different region of support: 9×9, 17×17, and 35×35. In order to obtain a large training set, we have collected 17 video sequences of 30 frames each from outside the standard MPEG sequences[1]. The MPEG sequences are kept for the test phase, because they can be easily compared to other techniques for which simulation results are available in the literature. We also apply a threshold to the energy of the residual to control the bit-rate during learning. This approach is motivated by the fact that each frame in each sequence has a different energy, and when this energy is low, the algorithm primarily encodes noise. The threshold is set empirically, in order to match as precisely as possible the bit-rates suggested for the different MPEG sequences. Finally, usage statistics are used to reduce the size of the learned dictionary from $3 \times 472 = 1416$ down to 472. Results obtained with this learning scheme and the given simulation strategy are presented in the following section and compared to the results obtained with the reference dictionary, h30.

4. RESULTS

A subset of the learned dictionary is shown in Fig. 1. After statistical pruning, it contains 116 functions from the 35×35 dictionary (24.47 %), 169 functions from the 17×17 dictionary (35.65 %), and 189 functions from the 9×9 dictionary (39.88 %). Most of these functions have therefore a small region of support. In general, they are well centered, oriented, limited in size, and modulated. We therefore expect that the learned functions can be easily and efficiently approximated with functions of low complexity for fast implementation [8]. The fact that the learned functions have a coherent structure is an encouraging result, given that learning schemes providing functions of such a “quality” are difficult to establish, in computer vision applications in general [9].

The ranked usage statistics of all function in h30 and in the learned dictionary are plotted in Fig. 2. These distributions show that the learned dictionary gives almost equal importance to all functions. In that sense, our learning scheme is very efficient.

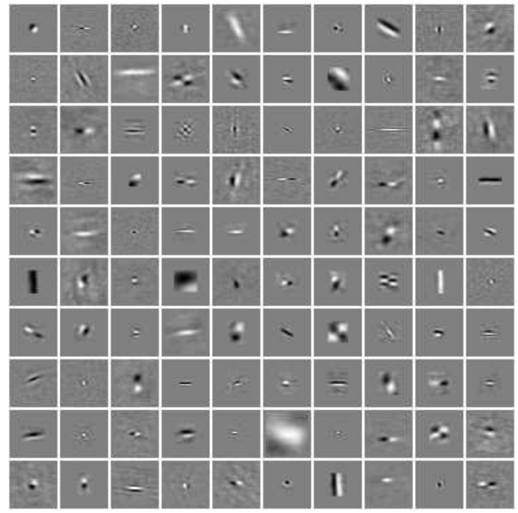

Fig. 1. Subset of learned dictionary.

sequence	kbps	fps	h30[dB]	new[dB]	gain[dB]
container	17.3	10	33.22	33.46	0.24
container	35.2	30	33.38	33.8	0.42
t. tennis	47.6	10	33.16	33.27	0.11
t. tennis	59.5	30	33.28	33.55	0.27
foreman	62.5	10	32.89	33.07	0.18
coast	81.5	10	31.94	32.19	0.25
foreman	112.6	30	33.05	33.49	0.44
coast	156.0	30	32.11	32.59	0.48
mobile	313.3	30	27.87	28.53	0.66
stefan	315.1	30	29.74	30.3	0.56

Table 1. Mean PSNR performances for Y component.

The learned dictionary is evaluated with 6 QCIF test sequences: Foreman, Coast, Table tennis, Container, Mobile, and Stefan. In all simulations, in order to guarantee similar bit-rate between h30 and our newly designed dictionary, we use the bit trace corresponding to h30 runs to control the bit-rate of our designed dictionary, even though this could potentially lower its performance. The performance results are summarized in Table 1 and the PSNR plot for Mobile is shown in Fig. 3. These results show that learning new dictionaries improves PSNR performances especially at higher bit-rates, since at low bit-rates most of the bit budget is spent on the motion vectors.

The time required to run a complete set of learning simulations is around 4 days on a Silicon Graphics Onyx computer. The reasons are (a) the large number of patterns ex-

[1]The reason for focusing on short sequences is to use as many different sequences as possible while maintaining the total number of training patterns at a reasonable level, in our case around 120,000.

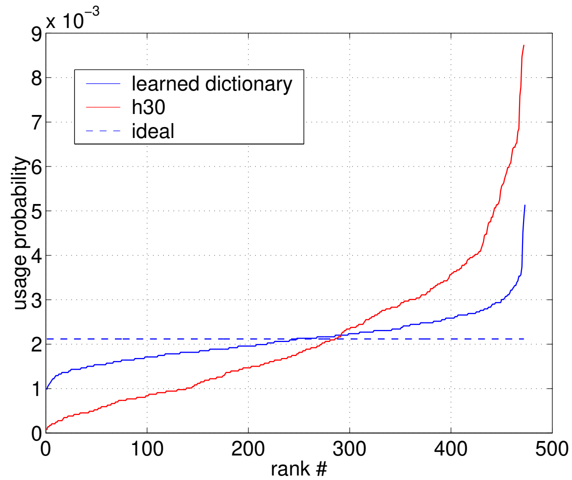

Fig. 2. Ranked usage statistics of dictionary functions.

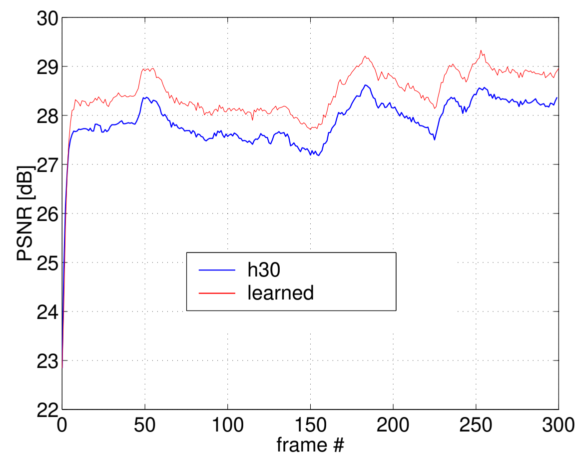

Fig. 3. Test sequence Mobile encoded with learned dictionary.

tracted from the training sequences for the learning phase, i.e. around 120,000 patterns of size 35×35, (b) the successive training cycles necessary to prune the original dictionary from 1416 to 472 functions, and (c) the computation of the Huffman codes for the different atom parameters, such as position, amplitude, and label. The test phase requires additional computation time as well.

5. CONCLUSIONS & FUTURE EXTENSIONS

In this paper we present a learning scheme for video coding based on *matching pursuits* (MP). The extraction of training patterns from the motion residuals, the learning scheme based on vector quantization (VQ), the pruning and finally the testing are explained in detail throughout the different sections. With our technique, we obtain encouraging results for bit-rates above 100 kbps.

A possible direction for future research is to design dictionaries for different classes of video sequences such as animations, high motion sports, head and shoulders, etc. The approximation of the dictionary functions that leads to an efficient implementation [8] is also a very important issue, and will be investigated in the future.

6. REFERENCES

[1] R. Neff and A. Zakhor, "Very low bit-rate video coding based on matching pursuits," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 158–171, 1997.

[2] O. Al-Shaykh, E. Miloslavsky, T. Nomura, R. Neff, and A. Zakhor, "Video compression using matching pursuits," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 1, pp. 123–143, 1999.

[3] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[4] C. De Vleeschouwer and B. Macq, "New dictionaries for matching pursuits video coding," in *Proceedings of ICIP '98*, 1998, vol. 1, pp. 764–768.

[5] D. W. Redmill, D. R. Bull, and P. Czerepiński, "Video coding using a fast non-separable matching pursuits algorithm," in *Proceedings of ICIP '98*, 1998, vol. 1, pp. 769–773.

[6] Y.-T. Chou, W.-L. Hwang, and Ch.-L. Huang, "Very low-bit video coding based on gain-shape VQ and matching pursuits," in *Proceedings of ICIP '99*, 1999, vol. 2, pp. 76–80.

[7] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[8] R. Neff and A. Zakhor, "Dictionary approximation for matching pursuit video coding," in *Proceedings of ICIP' 2000*, 2000.

[9] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.