

# Dictionary Design for Matching Pursuit and Application to Motion Compensated Video Coding

Philippe Schmid-Saugeon and Avidesh Zakhor

Berkeley, CA

philippe.schmid@ieee.org

## Abstract

We present a new algorithm for *matching pursuit* (MP) dictionary design. This technique uses existing *vector-quantization* (VQ) design techniques and an inner-product based distortion measure to learn functions from a set of training patterns. While this scheme can be applied to many MP applications, we focus on *motion compensated video coding*. Given a set of training sequences, data is extracted from the high energy packets of the motion compensated frames. Dictionaries with different regions of support are trained, pruned, and finally evaluated on MPEG test sequences. We find that for high bit-rate QCIF sequences we can achieve improvements of up to 0.66 dB with respect to conventional MP with separable Gabor functions.

## Index Terms

video coding, matching pursuit, dictionary design, vector-quantization

## I. INTRODUCTION

Matching pursuit (MP) based video codecs have shown to be competitive with hybrid motion-compensated block-based discrete cosine transform (DCT) codecs [1], [2], [3], [4]. MP was re-introduced from statistics to the signal processing community by Mallat and Zhang in 1993 [5]. MP expands a signal using 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. The use of MP to encode the motion compensated frames was originally proposed in [1]. Details on MP derivation and its mathematical properties can be found in [5], [6].

In most MP based video codecs, separable Gabor functions are used to encode the motion compensated frames. This leads to a fast implementation of the MP algorithm, but has a number of drawbacks: (a) there is no orientation information, and (b) Gabor atoms have the tendency to introduce small oscillations, especially when the number of atoms used to encode the signal is small, i.e. at very low bit-rates. A number of modifications have been proposed, without noticeable improvement. In [7], the authors propose to use a sub-band dictionary, which significantly reduces the computational cost without loss in performance. In [8], the authors propose a fast matching pursuit algorithm that uses non-separable dictionary functions. A bank of filters is used to produce  $N$  filtered signals from the input signal. The algorithm updates these signals after each iteration, obviating the need to repeat the filtering. However, this technique requires the storage of the  $N$  filtered signals, even if atoms are given as a linear combination of a small number of basis functions [8]. With this approach, improvements over non-separable dictionaries are less than 0.5 dB. Goodwin [9] has used damped sinusoids to model signals with transient behavior, and has shown that expansions using this kind of dictionary can be efficiently derived using simple recursive filter banks. However, the extension to two-dimensional signals is not straightforward. Finally, Chou et al. [10] have used gain-shape vector quantization to learn new dictionaries. However, their learning scheme does not take advantage of specific characteristics of MP.

Other techniques, such as the one proposed by Olshausen [11] might be used to learn functions from natural patterns; this approach is especially interesting because it embeds characteristics inspired by the human visual system. However, since it uses linear combinations of basis functions, it can not be adapted to MP applications.

In this paper, we propose a scheme to learn an MP dictionary from the motion compensated frames obtained from a set of training sequences. To begin with, the learning scheme must be adapted to MP applications. As such, we use *vector quantization* (VQ) [12] with a distortion measure based on the inner-product, as it is used in MP. Convergence is improved by using a decision rule that allows for removing small partitions and splitting larger partitions to keep the number of dictionary functions constant. Three dictionaries with different regions of support are learned, and usage statistics are used to reduce the number of functions. Huffman codes are then

computed from the usage statistics, and finally the new dictionary is evaluated on a set of test sequences.

This paper is organized as follows: Section II provides a short overview of matching pursuit. The proposed learning scheme is presented in Section III; results obtained for two different simulation scenarios are given in Section IV. Finally, conclusions are drawn in Section V.

## II. MATCHING PURSUIT

MP decomposes any signal  $f$  into a linear expansion of waveforms called hereafter *atom*. These *normalized* functions are selected from a *redundant*, i.e. over-complete, dictionary:

$$\mathbb{D} = \{g_\gamma\}_{\gamma \in \Gamma} , \quad (1)$$

where  $\Gamma = \{1, \dots, N\}$  is the set of all indices, and  $N$  is the dictionary size. MP is an *iterative* process producing at each iteration a new *residual* signal, which is then used as input to the next iteration. Initially, the residual is equal to the original signal. The residual at iteration  $m + 1$  is computed using the following equations:

$$R^{m+1}f = R^m f - \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m} , \quad (2)$$

$$R^0 f = f , \quad (3)$$

where  $R^m f$  is the residual at iteration  $m$ ,  $\langle \cdot, \cdot \rangle$  is the inner product, and  $g_{\gamma_m} \in \mathbb{D}$  the function whose inner product with the residual  $R^m f$  is at a maximum:

$$|\langle R^m f, g_{\gamma_m} \rangle| = \sup_{g_\gamma \in \mathbb{D}} |\langle R^m f, g_\gamma \rangle| . \quad (4)$$

After  $M$  iterations, the decomposed signal can be written in terms of the successively matched atoms as:

$$f = \sum_{m=0}^{M-1} \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m} + R^M f . \quad (5)$$

The inner product between the residual at iteration  $m + 1$  and the atom at iteration  $m$  is given by:

$$\begin{aligned} \langle R^{m+1} f, g_{\gamma_m} \rangle &= \langle R^m f - \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m}, g_{\gamma_m} \rangle \\ &= \langle R^m f, g_{\gamma_m} \rangle - \langle R^m f, g_{\gamma_m} \rangle \langle g_{\gamma_m}, g_{\gamma_m} \rangle \\ &= 0 , \end{aligned} \quad (6)$$

which means that the vectors are orthogonal to each other. The energy of the signal can therefore be written as the sum of the different contributions:

$$\|f\|^2 = \sum_{m=0}^{M-1} |\langle R^m f, g_{\gamma_m} \rangle|^2 + \|R^M f\|^2 . \quad (7)$$

For large signals, the computation time remains prohibitive and solutions must be found to speed up the encoding. In our case,  $f$  is the motion compensation error. It is a sparse signal for which the matching process can be confined to high-energy regions. These regions are detected by splitting the residual image into smaller blocks, and by computing the energy of each block. The matching process is then limited to the regions surrounding the high energy blocks [2].

The number of MP iterations depends on the bit-rate, i.e. the higher the bit-rate the larger the number of atoms necessary to encode the motion compensation error. Thus, each motion compensation error frame is only approximated, and the encoding error  $R^M f$  is propagated through the succeeding frames.

### III. DICTIONARY DESIGN

#### A. Learning patterns with vector-quantization

Our proposed learning scheme is based on vector quantization (VQ) [12]. It is an iterative process that learns a predefined number of vectors, called hereafter *code-vectors*, from a set of input vectors, called hereafter *patterns*, according to some distortion measure. Each iteration is made of two processing steps:

- 1) Partition the pattern space: this is achieved by grouping patterns whose distortion with respect to a given code-vector is minimum.
- 2) Update the code-vectors: this is achieved by computing the vectors that minimize the sum of all distortions within the different partitions.

The algorithm ends when a predefined stopping criterion is met, such as a threshold on the overall distortion.

Matching pursuit uses the inner product to match the different dictionary functions to the residuals and to select the 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  $k$ ,  $\mathbb{X} = \{1, \dots, N\}$  be the set of all code-vector

indices, and  $n$  be the iteration number in the dictionary design process. The energy  $\omega_i$  of each pattern is computed before normalization. These values are later used during the code-vector update step. We define the following distortion measure between a normalized pattern  $\mathbf{x}_i \in \mathbb{S}$  and the  $j^{\text{th}}$  normalized code-vector  $\hat{\mathbf{x}}_{j,n}$ :

$$d_{\langle \cdot, \cdot \rangle}(\mathbf{x}_i, \hat{\mathbf{x}}_{j,n}) = 1 - |\langle \mathbf{x}_i, \hat{\mathbf{x}}_{j,n} \rangle|, \quad (8)$$

where  $\langle \cdot, \cdot \rangle$  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  $\hat{\mathbf{x}}_{j,n}$ :

$$\mathbb{S}_{j,n} = \{ \mathbf{x}_i \in \mathbb{S} \mid d_{\langle \cdot, \cdot \rangle}(\mathbf{x}_i, \hat{\mathbf{x}}_{j,n}) \leq d_{\langle \cdot, \cdot \rangle}(\mathbf{x}_i, \hat{\mathbf{x}}_{l,n}), \forall l \in \mathbb{X} \}, \quad (9)$$

and

$$\mathbb{S} = \bigcup_{j \in \mathbb{X}} \mathbb{S}_{j,n}, \quad (10)$$

$$\mathbb{S}_{j,n} \cap \mathbb{S}_{l,n} = \emptyset, \quad (11)$$

$\forall j \neq l$  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 weighted distortion  $\delta_{j,n}$  in  $\mathbb{S}_{j,n}$ :

$$\delta_{j,n} = \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i d_{\langle \cdot, \cdot \rangle}(\mathbf{x}_i, \hat{\mathbf{x}}_{j,n+1}) \leq \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i d_{\langle \cdot, \cdot \rangle}(\mathbf{x}_i, \mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^k \quad (12)$$

The minimization of Eq. 12 with the distortion measure defined in Eq. 8 requires the use of Lagrange multipliers. 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. 8:

$$\begin{aligned} d_{L_2}(\mathbf{x}_i, \hat{\mathbf{x}}_{j,n}) &= \|\hat{\mathbf{x}}_{j,n} - \mathbf{x}_i\|^2 \\ &= (\hat{\mathbf{x}}_{j,n} - \mathbf{x}_i) \cdot (\hat{\mathbf{x}}_{j,n} - \mathbf{x}_i)^T \\ &= 2 - 2\hat{\mathbf{x}}_{j,n} \cdot \mathbf{x}_i^T \\ &= 2(1 - \langle \mathbf{x}_i, \hat{\mathbf{x}}_{j,n} \rangle), \end{aligned} \quad (13)$$

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  $-\mathbf{x}_i$ , and use the one resulting in a positive

inner product in Eq. 13. This is possible because Eq. 8 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 positive and negative inner products with  $\hat{\mathbf{x}}_{j,n}$ , respectively:

$$\mathbb{S}_{j,n}^{(+)} \cup \mathbb{S}_{j,n}^{(-)} = \mathbb{S}_{j,n} , \quad (14)$$

$$\mathbb{S}_{j,n}^{(+)} \cap \mathbb{S}_{j,n}^{(-)} = \emptyset . \quad (15)$$

Once both subsets are computed, we can use Eq. 13 instead of Eq. 8 by taking the negative value of the inner product for each pattern in  $\mathbb{S}_{j,n}^{(-)}$ . Using Lagrange multipliers, we have shown in the Appendix that the minimization of Eq. 12 with the distortion measure defined by Eq. 13 leads to the following weighted average update equation:

$$\hat{\mathbf{x}}_{j,n+1} = \frac{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i \mathbf{x}_i - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i} . \quad (16)$$

More weight is given to high energy patterns in Eq. 16, since it is essential to first encode high energy structures present in the motion compensation error. The code-vectors are normalized after being updated.

The derivation of this update equation is based on the assumption that for each partition the sign of the inner product between the code-vector and the patterns will not change after the code-vector is updated. This is of course not exactly the case in practice. However, because the centroids change slowly from one iteration to another, this assumption breaks down only for a small sub-set of the patterns, and only affects vectors whose inner product with the code-vector is close to zero, i.e. those for which a sign change can happen. The validity of our assumptions in practice will be illustrated in Section IV, and the full derivation of Eq. 16 is given in Appendix.

In order to prevent the VQ algorithm from converging to a local minimum, it is necessary to modify the above two-step scheme. Different improvements have been proposed in the literature, such as coupling stochastic relaxation methods with VQ [13]. In [14], the authors propose a deterministic annealing approach. Both approaches are formulated within a probabilistic framework and lead to complex and time-consuming processes, even though they are faster than techniques based on simulated annealing. The use of fuzzy sets theory for VQ has been proposed by Karayianis et al. [15], [16]. They have successfully applied it to compression problems, but their approach is still very expensive in terms of computation time. In order to achieve a fair balance

between computation time and reliability, we set a time-decreasing threshold on the partition size in order to decide which partitions should be suppressed. 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. We use the following exponential threshold function in our simulations:

$$\Omega_{\text{thresh}} = \frac{\Omega}{N} \exp \left\{ -\frac{M}{M_0} \right\}, \quad (17)$$

where  $M$  is the iteration number,  $M_0$  is a constant scalar that controls the convergence rate,  $N$  is the number of code-vectors, and  $\Omega$  is the weighted size of the pattern space:

$$\Omega = \sum_{i=1}^n \omega_i. \quad (18)$$

In our simulations we set  $M_0 = 20$ , and  $\Omega_{\text{thresh}}$  is only used every four iterations in order to allow the system to stabilize in the neighborhood of a local minimum:

$$\Omega_{\text{thresh}} = \begin{cases} \frac{\Omega}{N} \exp \left\{ -\frac{M}{M_0} \right\} & \text{if } M \bmod 4 = 0, \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

While this approach is of low complexity, we have shown it to be robust, and to lead to near-optimal results.

### *B. Learning cycle for motion compensated video coders*

The extraction of training patterns from the motion residuals is an important issue. The entire residual cannot be learned by our proposed scheme since high energy pockets are sparsely distributed. Only regions in the residual where one or several dictionary functions are matched can be 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* [17], 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 previously been matched, their content is influenced by the dictionary itself. In addition to that, residuals in

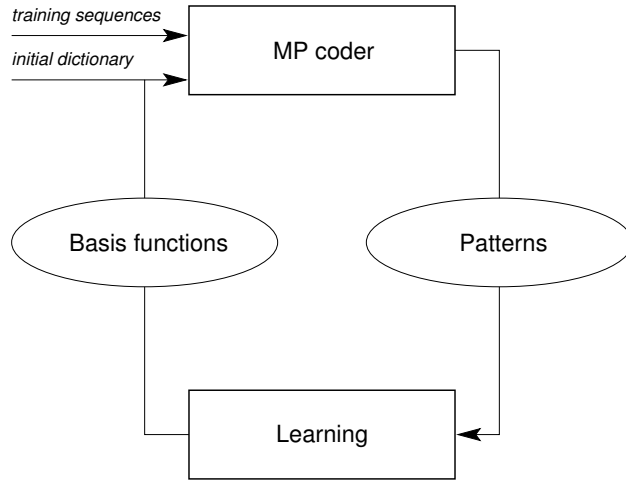


Fig. 1. Training cycle for learning basis functions.

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 to address this, we must repeat the *pattern extraction-dictionary design* cycle several times, using the successively learned dictionaries to extract a new set of patterns. This concept is shown in Figure 1. This is a time consuming process, and we have experienced that even running only one cycle takes a long time. Moreover, we have run several cycles in some of our simulations and have found no noticeable difference between successive cycles. The results shown in this paper are obtained with only one learning cycle. Further work is needed to examine the effect of running multiple cycles.

Finally, note that once a new dictionary is 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.

#### IV. SIMULATIONS AND RESULTS

All sequences used to train and test the dictionaries are in *QCIF*. In order to obtain a large training set, we have collected 17 high motion sequences of 30 frames from outside the standard MPEG sequences. We call them *anonymous* sequences because they do not belong to any standard



group of test sequences. The purpose of using such short sequences is (a) to increase the diversity of the patterns needed in the learning phase while maintaining their number at a reasonable level, in our case more than 100,000, and (b) to reduce the influence of the initial dictionary on the pattern extraction due to the propagation of the encoding error. The standard MPEG sequences are kept for the test phase, because they can be easily compared to other techniques for which simulation results are readily available in the literature.

We learn three different dictionaries, each one having a different region of support:  $9 \times 9$ ,  $17 \times 17$ , and  $35 \times 35$  pixels. A threshold is applied to the energy of the motion residual to control the bit-rate. This approach is motivated by the fact that the energy of the residual signal varies for each sequence and each frame, and that at low energy values we encode mainly noise. A unique threshold is empirically chosen for all simulations, in order to match the bit-rates suggested for the different MPEG sequences to enable fair comparison. The *foreman*, *coast*, *table tennis*, *container*, *mobile*, and *stefan* sequences are used to evaluate the performance of the learned dictionary.

Once a new dictionary is learned, a Huffman code is generated, based on the atom statistics obtained during the encoding of the training sequences. Two techniques are used to reduce the size of the dictionaries: the first one is based on the usage statistics and allows reduction of the number of functions from  $3 \times 472 = 1416$  down to 472, which is the size of our reference dictionary *h30* [17]. A second approach is based on the pair-wise cross-correlation of the different dictionary functions. It allows further pruning of the final dictionary while keeping the performance almost at the same level. The idea is simply to remove similar functions, which typically occur in dense regions of the pattern space for which the VQ scheme learns several similar code vectors.

The performances are summarized in Table I. Generally speaking, the learned dictionary outperforms *h30* at higher bit-rates. This is because at higher bit-rates a larger portion of the bits is devoted to texture coding. At low bit-rates, most bit budget is spent on motion and learning dictionaries does not help improving the performance. As an example, at bit rates around 300 kbps, improvements are 0.66 and 0.56 dB for *mobile* and *stefan*, respectively.

After statistical pruning, the new dictionary contains 116 functions from the  $35 \times 35$  dictionary (24.47 %), 169 functions from the  $17 \times 17$  dictionary (35.65 %), and 189 functions from the  $9 \times 9$  dictionary (39.88 %). Most of these functions have therefore a small region of support.

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

TABLE I

MEAN PSNR PERFORMANCE FOR Y COMPONENT.

Usage statistics are collected when encoding the different test sequences. Dictionary functions are then sorted in the order of increasing usage. Figure 2 shows the ranked usage statistics obtained for the new dictionary. As seen, the distribution is much more uniform than for *h30*. This is indeed a desirable property for the learned dictionary in the sense that all its functions should be of equal importance. Functions that are rarely used, if removed from the dictionary, would not, on average, significantly reduce the quality of encoded sequences. If all functions are, on average, equally used, they equally contribute to the encoding process and they are equally important. When the number of functions in the dictionary and hence the time necessary for the matching process must be kept as low as possible, this property is indeed desirable. Figure 2 illustrates that with the newly designed dictionary we are closer to this ideal situation than with dictionary *h30*.

Figure 3 plots the mean distortion at each iteration when learning the dictionary with the  $17 \times 17$  region of support from the anonymous sequences. As expected, it is a monotonically decreasing function, except when partitions are split. Iterations where splitting occur are marked with a  $\circ$ . As underlined in section III, the update equation 16 is only valid if for all code-vectors the sign of the inner-products before and after they are updated remain unchanged for all the patterns that belong to their partition. In our simulations, we found the number of code vectors which violate this constraint to be non zero only at the first iteration. This is actually due to the

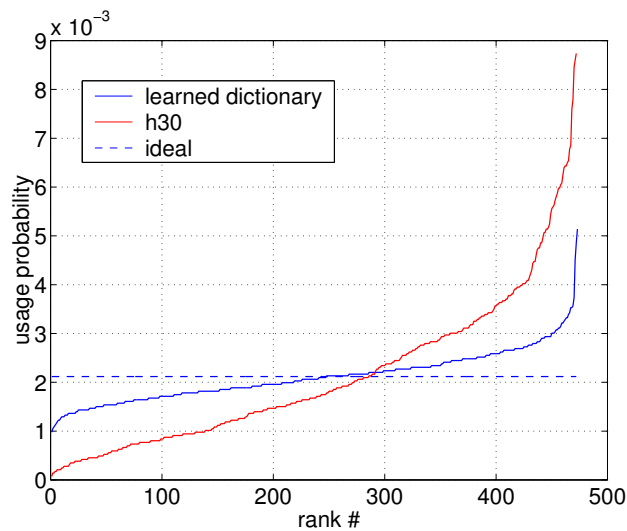


Fig. 2. Ranked usage statistics of dictionary functions (B).

fact that code-vectors are initialized randomly.

A subset of the new dictionary is shown in Figure 4. As seen, the learned functions have a coherent structure: they are centered, oriented, limited in size, and modulated. The Fourier transforms of these functions are shown in Figure 5. We expect the learned functions to be easily and efficiently approximated with functions of low complexity for a fast implementation [17], [18]. The fact that the learned dictionaries have a coherent structure is an encouraging result, knowing that learning schemes providing functions of such a “quality” are generally difficult to establish in computer vision applications [19].

In all simulations, the rate control is matched to the runs obtained by encoding with *h30*, which in turn is achieved by using an energy threshold. As such, this is inherently biased towards *h30*, and might limit the performance of the learned dictionary. However, this is necessary in order to keep the bit-rates equal and hence the comparisons meaningful. The time required to run a complete set of simulations with training and testing is on the order of several days on a Silicon Graphics Onyx computer with eight processors. The main reason is that the amount of data extracted from the training sequences for the learning phase is huge: more than 100’000 patterns of size  $35 \times 35$ . However, this is a one-time cost that should only be borne once. Once the new

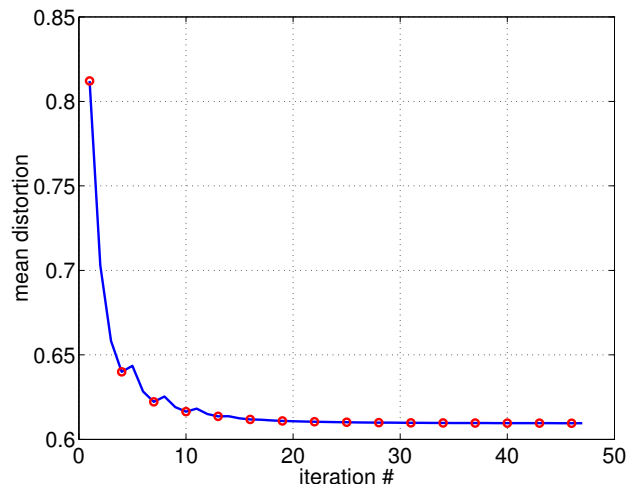


Fig. 3. Evolution of mean distortion during learning (B).

dictionary is designed, it can be used over and over in many encoding/decoding scenarios. In addition to that, the dictionaries can be approximated with a series of elementary functions as described in [18], and hence further speed ups can be expected in the future.

## V. CONCLUSIONS

In this paper we present a dictionary design technique for video codecs based on *matching pursuit*. A learning scheme based on *vector-quantization* has been developed for this purpose. Learning new dictionaries requires many time-consuming processing steps in order to extract training patterns from a set of sequences, compute usage statistics to prune the learned dictionaries, and compute the Huffman codes.

Our final learning scheme has been to use three different fixed regions of support for patterns extracted from the motion corrected frames. An energy threshold has been used to set the number of atoms encoded for each frame, thus avoiding to encode noise. Once the three dictionaries are computed, a pruning based on the usage statistics is performed, and finally the Huffman codes are computed. Examples are given for sequences in QCIF format. For bit-rates above 100 kbps, we have obtained improvements of up to 0.66 dB with respect to conventional MP with separable Gabor functions. We found that learning dictionaries for low-motion sequences does not allow

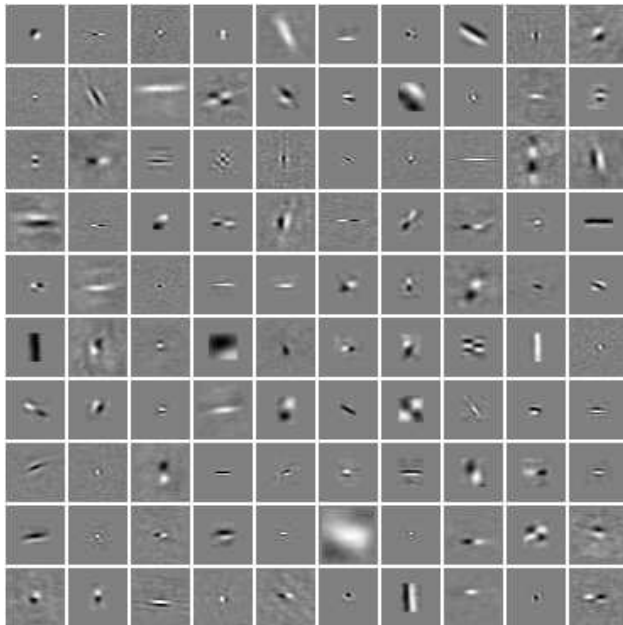


Fig. 4. Subset of dictionary functions (B).

for significant improvements in performances.

Future work involves learning dictionaries for different categories of sequences and the approximation with elementary functions that can be implemented efficiently [18]. The definition of specific characteristics that can be used to select an appropriate dictionary for a given sequence would also be of great interest. Finally, additional simulations will help understanding the effect of iterative dictionary learning, where a learned dictionary is used as the reference dictionary in the pattern extraction phase of the next iteration.

## VI. APPENDIX

The minimization of Eq. 12 with the  $L_2$ -norm distortion measure given by Eq. 13 is obtained by setting to zero all partial derivatives:

$$\frac{\partial}{\partial \hat{x}_{l,j}} \left( \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i d_{L_2}(\mathbf{x}_i, \hat{\mathbf{x}}_{j,n+1}) \right) = 2 \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i (\hat{x}_{l,j} - x_{l,i}) = 0, \quad (20)$$

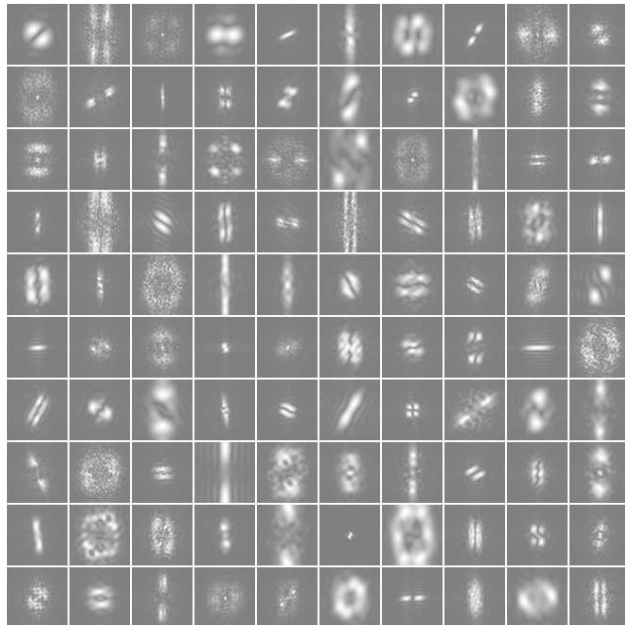


Fig. 5. Fourier transform of functions shown in Figure 4.

where

$$\hat{\mathbf{x}}_{j,n+1} = \begin{pmatrix} \hat{x}_{1,j} & \cdots & \hat{x}_{k,j} \end{pmatrix}, \quad (21)$$

$$\mathbf{x}_i = \begin{pmatrix} x_{1,i} & \cdots & x_{k,i} \end{pmatrix}. \quad (22)$$

From this set of  $k$  equations we obtain the following update formula:

$$\hat{\mathbf{x}}_{j,n+1} = \frac{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i}. \quad (23)$$

In this update equation we assume that all inner products are positive. If we take into account the sign test we did during the partition phase, then the update equation becomes:

$$\hat{\mathbf{x}}_{j,n+1} = \frac{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i \mathbf{x}_i - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i}, \quad (24)$$

which is Eq. 16. Note that the updated centroids must be normalized after they are computed.

To verify that the minimization of Eq. 12 with the distortion measure given by Eq. 8 under the constraint that the updated centroids have unit norm is equivalent to the above result, we need to use *Lagrange multipliers*. The initial set of  $k + 1$  equations that must be solved is:

$$\frac{\partial}{\partial \hat{x}_{l,j}} \left( \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i (1 - |\langle \mathbf{x}_i, \hat{\mathbf{x}}_{j,n+1} \rangle|) \right) + \lambda \frac{\partial}{\partial \hat{x}_{l,j}} \left( -1 + \sum_{m=1}^k \hat{x}_{m,j}^2 \right) = 0, \quad (25)$$

$$-1 + \sum_{m=1}^k \hat{x}_{m,j}^2 = 0, \quad (26)$$

which, when using the subsets  $\mathbb{S}_{j,n}^{(+)}$  and  $\mathbb{S}_{j,n}^{(-)}$ , become

$$\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i x_{l,i} - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i x_{l,i} + 2\lambda \hat{x}_{l,j} = 0, \quad (27)$$

$$-1 + \sum_{m=1}^k \hat{x}_{m,j}^2 = 0, \quad (28)$$

for  $l = 1, \dots, k$ . Using simple calculus the following solution is obtained:

$$\hat{x}_{1,j} = \pm \sqrt{\frac{\alpha_1^2}{\sum_{m=1}^k \alpha_m^2}}, \quad (29)$$

$$\hat{x}_{2,j} = \frac{\alpha_2}{\alpha_1} \hat{x}_{1,j}, \quad (30)$$

⋮

$$\hat{x}_{k,j} = \frac{\alpha_k}{\alpha_1} \hat{x}_{1,j}, \quad (31)$$

where

$$\alpha_l = \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i x_{l,i} - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i x_{l,i}. \quad (32)$$

Since the sign of  $\hat{\mathbf{x}}_{j,n+1}$  does not affect the distortion measure, i.e. the same distortion value is obtained taking  $\hat{\mathbf{x}}_{j,n+1}$  or  $-\hat{\mathbf{x}}_{j,n+1}$ , we arbitrarily choose  $\hat{x}_{1,j} > 0$ . Knowing that the norm of the updated centroid when using Eq. 24 is given by:

$$\frac{1}{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i} \sqrt{\sum_{m=1}^k \alpha_m^2} = \left\| \frac{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i \mathbf{x}_i - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i} \right\|, \quad (33)$$

and that

$$\left| \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i x_{1,i} - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i x_{1,i} \right| = \text{sign}(\alpha_1) \left( \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i x_{1,i} - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i x_{1,i} \right), \quad (34)$$

the following result is obtained:

$$\hat{\mathbf{x}}_{j,n+1} = \text{sign}(\alpha_1) \frac{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i \mathbf{x}_i - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i} \left\| \frac{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(+)}} \omega_i \mathbf{x}_i - \sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}^{(-)}} \omega_i \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathbb{S}_{j,n}} \omega_i} \right\|^{-1}, \quad (35)$$

which is the normalized version of  $\hat{\mathbf{x}}_{j,n+1}$  computed with Eq. 16.  $\text{sign}(\alpha_1)$  affects all components of  $\hat{\mathbf{x}}_{j,n+1}$  and can therefore be disregarded.

## REFERENCES

- [1] R. Neff, A. Zakhor, and M. Vetterli. Very low bit rate video coding using matching pursuits. In *Proceedings of the SPIE*, volume 2308, pages 47–60, 1994.
- [2] R. Neff and A. Zakhor. Very low bit-rate video coding based on matching pursuits. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):158–171, February 1997.
- [3] M. R. Banham and J. C. Brailean. A selective update approach to matching pursuits video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):119–129, February 1997.
- [4] R. Neff, T. Nomura, and A. Zakhor. Decoder complexity and performance comparison of matching pursuit and dct-based mpeg-4 video codecs. In *Proceedings of ICIP '98*, volume 1, pages 783–787, 1998.
- [5] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, December 1993.
- [6] S. Mallat. *A Wavelet tour of signal processing*. Academic Press, 1998.
- [7] C. De Vleeschouwer and B. Macq. New dictionaries for matching pursuits video coding. In *Proceedings of ICIP '98*, volume 1, pages 764–768, 1998.
- [8] D. W. Redmill, D. R. Bull, and P. Czerepiński. Video coding using a fast non-separable matching pursuits algorithm. In *Proceedings of ICIP '98*, volume 1, pages 769–773, 1998.
- [9] M. M. Goodwin and M. Vetterli. Matching pursuit and atomic signal models based on recursive filter banks. *IEEE Transactions on Signal Processing*, 47(7):1890–1902, July 1999.
- [10] 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*, volume 2, pages 76–80, 1999.
- [11] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, 37(23):3311–3325, December 1997.
- [12] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, January 1980.
- [13] K. Zeger, J. Vaisey, and A. Gersho. Globally optimal vector quantizer design by stochastic relaxation. *IEEE Transactions on Signal Processing*, 40(2):310–322, February 1992.



- [14] K. Rose, E. Gurewitz, and G. C. Fox. Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory*, 38(4):1249–1257, July 1992.
- [15] N. B. Karayiannis and P.-I. Pai. Fuzzy algorithms for learning vector quantization. *IEEE Transactions on Neural Networks*, 7(5):1196–1211, September 1996.
- [16] N. B. Karayiannis, P.-I. Pai, and N. Zervos. Image compression based on fuzzy algorithms for learning vector quantization and wavelet image decomposition. *IEEE Transactions on Image Processing*, 7(8):1223–1230, August 1998.
- [17] R. Neff and A. Zakhor. Dictionary approximation for matching pursuit video coding. In *Proceedings of ICIP 2000*, volume 2, pages 828–31, 2000.
- [18] R. Neff and A. Zakhor. Matching pursuit video coding .I. dictionary approximation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(1):13–26, January 2002.
- [19] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, June 1996.