Video Coding Using a Fast Non-Separable Matching Pursuits Algorithm

David W. Redmill, David R. Bull and Przemysław Czerepiński
Image Communications Group, Centre for Communications Research,
University of Bristol, Bristol. BS8 1UB UK
Tel: [+44] 117 954 5195, Fax: [+44] 117 925 5265
Email: David.Redmill@Bristol.ac.uk Dave.Bull@Bristol.ac.uk

Abstract

This paper presents a fast matching pursuits algorithm. In addition to a fast implementation, the algorithm also allows the efficient use of non-separable dictionary basis functions. The use of non-separable components allows basis functions which provide a better match to diagonally orientated image features. In order to demonstrate the proposed method, a simple dictionary is developed and used. Even without any sophisticated entropy coding, the proposed system gives performance exceeding that of H.263.

1 Introduction

Matching pursuits was originally proposed by Mallat and Zhang [1], and later used for coding motion compensated video signals [2, 3]. Matching pursuits provides an effective and flexible method for low-bit rate coding.

The concept of matching pursuits lies in describing a signal $x(\mathbf{x})$ as a linear combination of a number of shifted and scaled basis functions (or atoms) $g_k(\mathbf{x})$ from a pre-defined dictionary $\mathcal{D}$.

$$\hat{x}(\mathbf{x}) = \sum_{i=1}^{M} \alpha_i g_{k_i}(\mathbf{x} + \mathbf{d}_i) \tag{1}$$

Thus a coded signal $\hat{x}(\mathbf{x})$ can be represented by a set of M code-words $c_i = \{k_i, \alpha_i, \mathbf{d}_i\}$ comprising a dictionary index k_i, magnitude α_i and shift $\mathbf{d}_i$. The decoder can reconstruct the coded signal $\hat{x}(\mathbf{x})$ quickly and simply using the above equation.

However the task of encoding, is considerably more difficult, since the encoder needs to determine a good set of code-words to efficiently represent the signal. For any given dictionary, an optimal solution to this problem is extremely difficult. Instead, an iterative procedure is used, which involves repeatedly finding the best code-word, providing the best match (maximum correlation) to the residual error $e(\mathbf{x}) = x(\mathbf{x}) - \hat{x}(\mathbf{x})$. Assuming that the dictionary functions are normalized, this can be represented as:

$$\max_{k,\mathbf{d}} \left| \sum_{\mathbf{x}} e(\mathbf{x}) g_k(\mathbf{x} + \mathbf{d}) \right| \tag{2}$$

The encoding algorithm is summarized in figure 1, and is seen to consist of an iterated loop containing a search stage, and an update stage. The number of code-words used is determined according to a termination criterion. This can be selected either to achieve a fixed target bit rate, or to maintain a desired quality.

Matching pursuits is particularly useful for data consisting of a sparse set of important features, such as motion compensated residual error signals. The advantages of the technique lies in the use of an oversampled set of basis functions which allows a description with many fewer code-words. The main disadvantage of matching pursuits is the computational complexity required at the encoder in order to repeatedly find the best basis function. The complexity of the search process is approximately $|\mathcal{D}||\mathcal{S}|\bar{n}$ multiplications and additions per atom, where $|\mathcal{D}|, |\mathcal{S}|$ and $\bar{n}$ are the size of the dictionary, the size of the search region and average number of non-zero points in the dictionary basis functions. For example the dictionary proposed in [2] contains 400 functions with an average of 210.25 non-zero points, giving a total complexity (for a non-separable implementation) of $84100|\mathcal{S}|$ multiplies and additions per atom. On top of this there is also the complexity of the update stage which is approximately $\bar{n}$ additions. Clearly the complexity of the search process significantly outweighs the complexity of the update stage.

Search complexity can be reduced in several ways.

0-8186-8821-1/98 $10.00 © 1998 IEEE

```
Calculate initial prediction $\hat{x}(\mathbf{x})$.
Calculate initial residual error $e(\mathbf{x}) = x(\mathbf{x}) - \hat{x}(\mathbf{x})$.
FOR each code-word $i$,
    Choose local search area $\mathcal{S}$.
    Find $\max_{k\in\mathcal{D},\mathbf{d}\in\mathcal{S}} |\sum_{\mathbf{x}} e(\mathbf{x})g_k(\mathbf{x}+\mathbf{d})|$.
    Update error $e(\mathbf{x}) = e(\mathbf{x}) - \alpha g_k(\mathbf{x}+\mathbf{d})$.
END
```

Figure 1: *Traditional Matching Pursuits Algorithm.*

Firstly the size of dictionary ($|\mathcal{D}|$) can be limited. However this can significantly reduce the performance, since the encoder may need to use more atoms to code the same feature. Secondly the encoder can use a dictionary with smaller basis functions (reducing $\bar{n}$). However, this limits the ability of the encoder to cope with large features. Thirdly the search area $|\mathcal{S}|$ can be reduced. This can be achieved by first finding the local maximum (or maxima) of some activity measure (such as local mean squared error) and then limiting the search to that vicinity. Finally, for multi-dimensional signals, the complexity can be further reduced by using separable basis functions [2]. Using this approach the complexity is reduced to approximately $(|\mathcal{D}| + \sqrt{|\mathcal{D}|})|\mathcal{S}|\sqrt{\bar{n}}$ adders and multipliers per atom (see [2] for a more complete analysis which yields a slightly larger figure). For example the dictionary proposed in [2] this comes to more than $6090|\mathcal{S}|$ multiplies and additions per atom. However, using only separable basis functions, limits the encoders ability to cope with diagonally orientated features.

2 Fast Matching Pursuits Algorithm

In this paper we shall present an alternative approach to constructing the dictionary, and performing the search, which both significantly simplifies the search process, and allows the use of non-separable basis functions. The approach is motivated by the observation that the correlation process of equation 2 is essentially a linear filtering operation. The aim is to remove this complex filtering operation from the matching pursuits loop, so that it only needs to be performed once, to initialize the system. To achieve this, a bank of N normalized filters $f_j(\mathbf{x})$ are used to produce N filtered error signals $e_j(\mathbf{x}) = \sum_{\mathbf{p}} f_j(\mathbf{x}-\mathbf{p})e(\mathbf{p})$. The algorithm updates these error signals after each atom, thereby avoiding having to repeat the filtering.

Choosing an N element dictionary $\mathcal{D}$ comprising the N filter functions, allows the search process to be written as $\max_{k,\mathbf{d}}|e_k(\mathbf{d})|$ (without the correlation). The update stage is then modified to update each of

```
Calculate initial prediction $\hat{x}(\mathbf{x})$.
Calculate initial residual error $e(\mathbf{x}) = x(\mathbf{x}) - \hat{x}(\mathbf{x})$.
Calculate $N$ filtered error signals:
    $e_j(\mathbf{x}) = \sum_{\mathbf{p}} f_j(\mathbf{x}-\mathbf{p})e(\mathbf{p})$
FOR each code-word $i$,
    Choose local search area $\mathcal{S}$.
    Find $\max_{k\in\mathcal{D},\mathbf{d}\in\mathcal{S}} |\sum_{n=1}^{n=n_k} \gamma_{k,n} f_{j_{k,n}}(\mathbf{d}+\mathbf{d}_{k,n})|$.
    Update the $N$ error signals:
        $e_i(\mathbf{x}) = e_i(\mathbf{x}) - \alpha\sum_{n=1}^{n=n_k} \gamma_{k,n} c_{i,j_{k,n}}(\mathbf{x}+\mathbf{d}+\mathbf{d}_{k,n})$.
END
```

Figure 2: *Fast Matching Pursuits Algorithm.*

the N filtered error signals. Since the basis functions are linear, this can be re-written as the addition of a pre-defined correlation function to each signal:

$$\begin{aligned} e_j(\mathbf{x}) &= \sum_{\mathbf{p}} f_j(\mathbf{x}-\mathbf{p})\left(e(\mathbf{p}) - \alpha g_k(\mathbf{p}+\mathbf{d})\right) \\ &= e_j(\mathbf{x}) - \alpha c_{j,k}(\mathbf{x}+\mathbf{d}) \end{aligned} \tag{3}$$

where $c_{j,k} = \sum_{\mathbf{p}} f_j(\mathbf{x}-\mathbf{p})f_k(\mathbf{p})$ is the pre-defined correlation function.

Using this approach, the search complexity is significantly reduced (by eliminating the correlations), at the expense of a complexity increase for the update stage. The proposed approach also requires an extra pre-filtering stage to generate the N filtered error signals $e_j(\mathbf{x})$. However, for effective compression, the dictionary size $|\mathcal{D}| = N$ needs to be relatively large, which implies a large memory requirement to store N filtered error signals, and the $N(N+1)/2$ correlation functions. Both the initial filtering, and the update stage also becomes increasingly complex as the number of filters is increased.

To overcome these problems, we propose a compromise, using a small set of N filters, and a much larger dictionary. Each dictionary function is composed from the weighted sum of n_k filtered samples:

$$g_k(\mathbf{x}) = \sum_{n=1}^{n=n_k} \gamma_{k,n} f_{j_{k,n}}(\mathbf{x}+\mathbf{d}_{k,n}) \tag{4}$$

where n_k is a small number (e.g. $n_k \leq 4$ for the results presented).

The search stage now requires a sum of n_k components for each candidate:

$$\max_{k,\mathbf{d}} \left| \sum_{n=1}^{n=n_k} \gamma_{k,n} f_{j_{k,n}}(\mathbf{d}+\mathbf{d}_{k,n}) \right| \tag{5}$$

Using this approach, the search complexity is reduced to approximately $|\mathcal{D}||\mathcal{S}|\bar{n}_k$ multiplications and additions (where $\bar{n}_k$ is the mean number of elements n_k

per dictionary function). Since $\bar{n}_k \ll \bar{n}$, the search complexity is significantly reduced. This is achieved at expense of a higher complexity for the update process, which requires Nn_k correlation functions to be subtracted. Note, that the update complexity is not a function of the search area or the dictionary size, but will vary according to which atom is chosen. The algorithm also requires the extra initialization stage comprising N filtering operations. For a large dictionary and search region $|\mathcal{D}||\mathcal{S}|$, the reduction in search complexity significantly out-weighs the increase in both initialization and update complexity. Figure 2 summarizes the proposed fast matching pursuits algorithm.

The searching process can be further simplified if the values $\gamma_{k,n}$ are chosen to provide simple multiplications. For example in the system presented below $\gamma_{k,n} = 1$ for all elements, and thus the search process requires no multiplications. The complexity can be reduced even further by calculating each dictionary function relative to a previous dictionary function. E.g. each function is determined by a previous function with one extra value:

$$g_k(\mathbf{x}) = g_{k'}(\mathbf{x}) + \gamma_k f_{j_k}(\mathbf{x} + \mathbf{d}_k) \qquad (6)$$

However, these approaches upset the relative normalization of the dictionary code-words. Thus, the maximization should find the best position for each dictionary code-word, and then normalize these maxima to find the overall maximum. Thus the search complexity becomes $|\mathcal{D}||\mathcal{S}|$ additions and comparisons (to find the $|\mathcal{D}|$ maxima), and a further $|\mathcal{D}|$ multiplications and comparisons to find the overall maximum.

3 Example Video Coding System

The system described above has been used for coding motion compensated frame difference images within a video codec. In order to provide a reasonable set of dictionary elements which represent a variety of features at differing scales, we have used a set of 8 filters with sizes (1x1, 3x3, 5x5, 7x7, 11x11, 15x15, 23x23, 31x31). To keep the filtering cost down, the filters are implemented as separable 1D filters. In order to avoid both blocking and ringing artifacts the filters chosen had a raised cosine shape.

A dictionary consisting of 128 functions has been used (as shown in figure 3). The dictionary functions have been chosen to approximate line segments with various widths. Each function contains at most 4 equal valued elements ($n_k \leq 4$), and differs by only 1 or 2 extra elements from other dictionary functions.

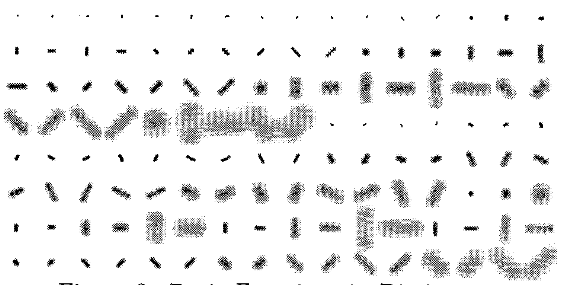
Figure 3: *Basis Functions in Dictionary.*

Thus, all of the above complexity saving methods can be employed. Note that this dictionary has been designed heuristically, with no attempt made to optimize it. We anticipate, that significant improvements may be achievable by optimizing and/or enlarging the dictionary. The search complexity for the proposed approach is found to be less than $2 \times 128|\mathcal{S}|$ adders, and a further 128 multipliers per atom. This compares favorably with the value of over $6090|\mathcal{S}|$ multiplies and additions per atom for the separable dictionary used in [2].

For the results presented in this paper we have used a simple coding system comprising fixed length codewords. This choice was motivated partly by the desire to develop a system which exhibits good error resilience, although this property is not considered here. The code comprises one bit for each 16x16 block determining if the block has a non-zero motion vector. If this bit is set, then the motion vector is coded as two 5 bit code-words. The matching pursuit code-words are considered to comprise 7 bits to specify the dictionary index, and 4 bits to specify the quantized gain α_i. Assuming that these code-words are randomly distributed over the image, the position information $\mathbf{d}_i$ can be efficiently coded using the error resilient positional code (ERPC) of [4]. The ERPC can be considered as using a comma code for each block. The total number of bits for a frame can now be determined as:

$$\begin{array}{ll} N_{\text{blk}} + 10N_{\text{vec}} & \text{(motion vectors)} \\ N_{\text{blk}} + 20N_{\text{mp}} & \text{(residual error)} \end{array}$$

where N_{blk} is the number of 16×16 blocks, N_{vec} is the number of non-zero motion vectors, and N_{mp} is the number of matching pursuit code-words. The value of 20 is a result of 7 bits for dictionary index, 4 bits for the magnitude, 8 bits specifying the position within a 16×16 block, and a comma bit specifying whether the block has another code-word.

The proposed system has been compared with both H.263, and alternative matching pursuits sys-

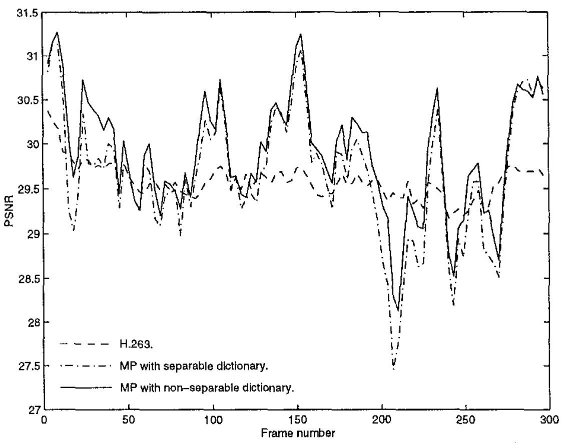

Figure 4: *Performance comparison for SQCIF 'Silent Voice' coded at 8.33 frames/s and 10 kbit/s.*

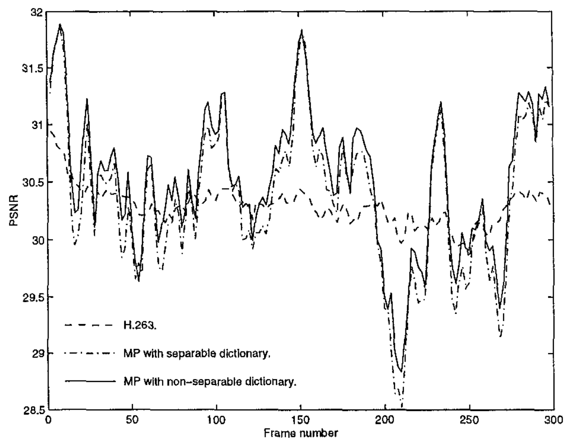

Figure 5: *Performance comparison for QCIF 'Silent Voice' coded at 12.5 frames/s and 20 kbit/s.*

tem based on the use of the separable dictionary of [2]. It should be noted that both these systems use more sophisticated entropy coding for both the motion vectors and the scale and index part of the matching pursuit code-words. Since the main interest here is the inter-frame coding, all three systems use the same intra coded first frame (obtained using H.263).

Figure 4 shows results obtained for coding a sub-QCIF sequence 'Silent Voice' at 8.33 frames/s and 10 kbit/s. The average PSNR values for the inter-coded frames were found to be: 29.89 dB for the fast non-separable matching pursuits system, 29.65 dB for the separable matching pursuits system, and 29.58 dB for H.263. Figure 5 shows results obtained for coding a QCIF sequence 'Silent Voice' at 12.5 frames/s and 20 kbit/s. The average PSNR values for the inter-coded frames were found to be: 30.51 dB for the fast non-separable matching pursuits system, 30.35 dB for the separable matching pursuits system, and 30.29 dB for H.263. Figure 6 shows the final frame (298) of the QCIF sequence coded using each system.

The results presented demonstrate that the fast non-separable matching pursuits system exhibits a significant performance improvement compared to both H.263 and the separable matching pursuits system. Visually the matching pursuits systems give much less apparent blocking artifacts (and better subjective quality) than the DCT based H.263 codec. Note that there are still some blocking artifacts which are a result of the block based motion compensation method. These could be reduced by using an overlapped motion compensation method. The matching pursuits systems both exhibit significant variations in both objective and subjective quality. This is primarily due to their fixed rate nature, while H.263 is inherently a variable rate system. If a constant quality were desired, then it is feasible to change the matching pursuits termination criterion to be quality dependent rather than bit-rate dependent. However, for a fixed rate, low delay system, matching pursuits offers significantly simpler rate control.

4 Conclusions

This paper has presented a fast matching pursuits algorithm. The use of this algorithm has allowed the use of non-separable diagonally orientated dictionary functions. The algorithm also allows a greater search area for a given complexity.

An example system has been presented with results exceeding those of both H.263, and a separable matching pursuits system. The results could be further improved by employing entropy coding for both the motion vectors and the shape and value parts of the matching pursuit code-words.

Since the dictionary used has been chosen heuristically without optimization, it is anticipated that, results may also be improved by optimizing and/or enlarging the dictionary.

The proposed approach also leads to a convenient compact method for defining dictionary basis functions (since each function consists of a few n_k elements). This in turn could be used within a system in which the dictionary is chosen by the encoder and transmitted prior to the signal.

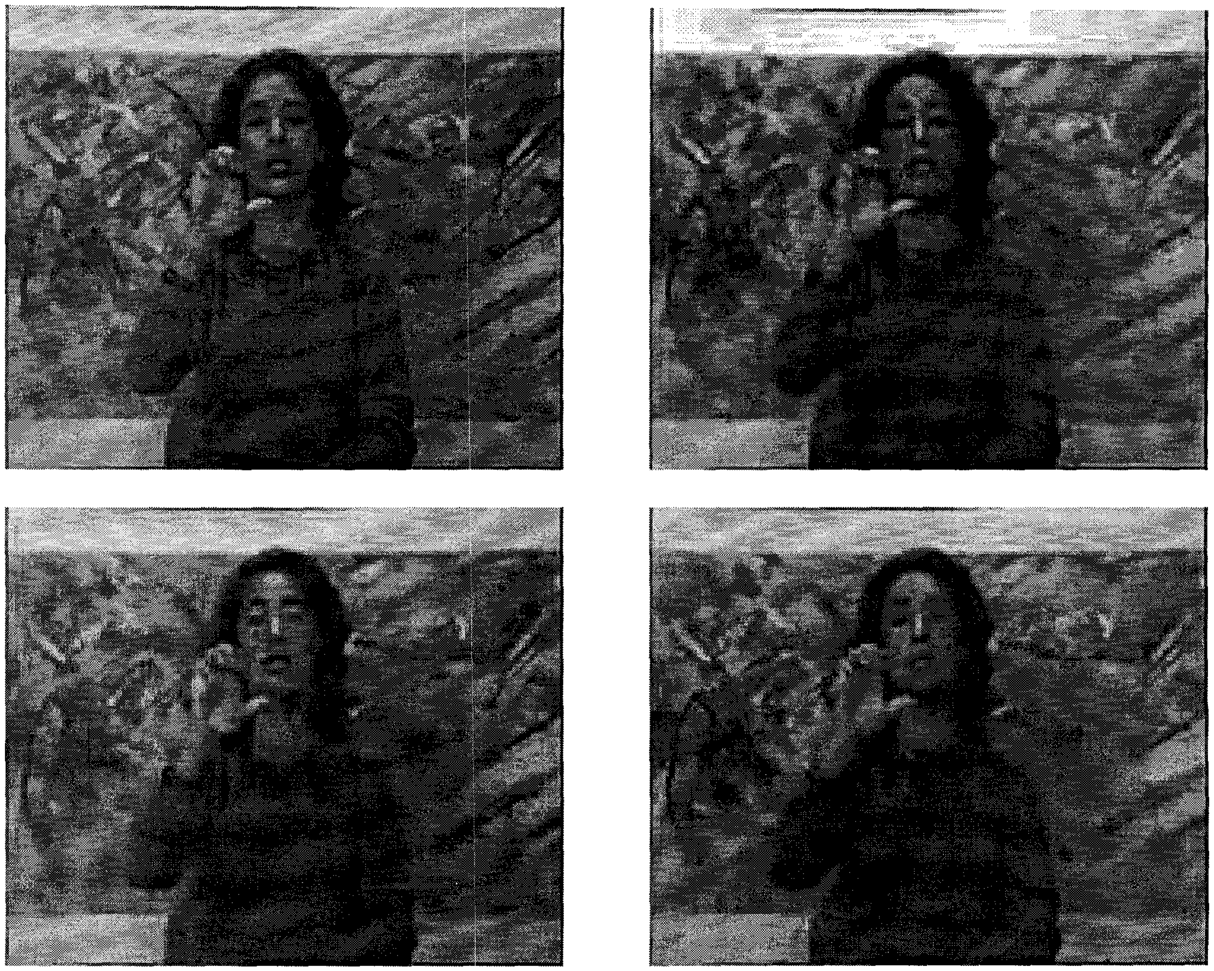

Figure 6: *Frame 298 of QCIF 'Silent Voice' coded at 12.5 frames/s and 20 kbit/s. a) Original. b) H.263 30.30 dB, c) Separable matching pursuits 31.16 dB d) Non-separable matching pursuits 31.17 dB.*

References

[1] S. Mallat and Z. Zhang. Matching pursuits with time frequency dictionaries. *IEEE Transactions on Signal Processing*, 41:3397–3415, 1993.

[2] R. Neff and A. Zakor. Very low bit-rate video coding based on matching pursuits. *IEEE Transactions on Circuits and Systems For Video Technology*, 7:158–171, 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*, 7:119–129, 1997.

[4] N. T. Cheng and N. G. Kingsbury. The ERPC: An efficient error-resilient technique for encoding positional information of sparse data. *IEEE Transactions on Communications*, 40:140–148, 1992.