Basis Optimisation for Multiscale Recurrent Pattern Coding

Nuno Rodrigues[1,2], Eduardo da Silva[3], Murilo de Carvalho[4], Sérgio de Faria[1,2], Vitor Silva[1,5]

[1]Instituto de Telecomunicações, Portugal; [2]ESTG, Instituto Politécnico de Leiria, Portugal; [3]PEE/COPPE/DEL/EE, Univ. Fed. do Rio de Janeiro, Brazil; [4]TET/CTC, Univ. Fed. Fluminense, Brazil; [5]DEEC, Univ. de Coimbra, Portugal.
e-mails: nuno.rodrigues@co.it.pt, eduardo@lps.ufrj.br, murilo@telecom.uff.br, sergio.faria, vitor.silva@co.it.pt.

Abstract - The Multidimensional Multiscale Parser (MMP) was recently proposed has a generic lossy data compression method. As a compression algorithm, MMP combines ideas from the adaptive vector quantisation and pattern matching theories with a new and important principle: approximate pattern matching with scales.

In spite of this, it will be shown here that the coding process of MMP can also be regarded as a decomposition of the original signal into a set of orthogonal, non overlapping, bases functions. In the original method, the initial dictionary uses a set of pulses with different amplitudes and scales, that correspond to the scaling function of the Haar's wavelet. The approximation of the input signal can be regarded has a combination of these pulses, using appropriate amplitudes, shifts and scales, that are determined by the MMP's rate-distortion controlled coding (or analysis) algorithm. The decoding (or synthesis) process uses the same basis functions to reconstruct the approximated signal.

The proposed problem is related to whether the use of base functions other than the Haar's scaling functions can be applied in the MMP's framework and what should be the characteristics of these base functions (orthogonality, perfect reconstruction, etc.), that allow their use and an increased performance of the method.

Keywords - Pattern matching, vector quantisation, signal decomposition, basis optimisation.

I. Introduction

The Multidimensional Multiscale Parser (MMP) [1] is a pattern matching based algorithm that combines the compression ideas from the work of Lempel-Ziv [2], patent in the dictionary updating procedure, with standard adaptive vector quantisation (VQ) algorithms [3] and with a new, important, principle: approximative pattern matching with scales.

MMP uses an adaptive dictionary of vectors to recurrently approximate variable-length blocks of input data. These variable-length blocks result from recursively parsing an original, fixed size, block of the message. Scaling transformations are used to adapt the size of each element of the dictionary into the size of the block segment that is being considered.

MMP has proved to have excellent adaptability characteristics, that allowed for excellent coding results in a wide range of data signals, like voice signals, electrocardiographic signal compression [4], stereoscopic images [5], digital images [1] [6] and video encoding [7].

In our work we focus mainly on the use of MMP for digital image coding, but the ideas presented here can be easily adapted for its use as a general lossy compression scheme for any data source. For image coding, MMP approximates the input image by using the concatenation of shifted and scale transformed (contracted or expanded) versions of the bi-dimensional blocks present in the dictionary. This process can be regarded as a decomposition of the original signal into a set of contiguous basis functions, that correspond to Haar's wavelets' scaling functions.

We are looking for insights that will allow for the use of different basis functions with MMP, that may provide for an increased compression performance.

In the next section we summarise the original MMP algorithm. Its application in image coding will be used has an example for this description. Section III will describe our interpretation of MMP has a signal decomposition method and we will present our problem on section IV.

II. The MMP Algorithm

MMP is based on approximations of data segments (in this case image blocks), using words of an adaptive dictionary $\mathcal{D}$ at different scales. For each block X^l in the image, the algorithm first searches the dictionary for the element S_i^l that minimises the Lagrangian cost function of the approximation. The superscript l means that the block X^l belongs to $level\ l$ of the segmentation tree. Square blocks, corresponding to even levels, are segmented into two vertical rectangles. Generally, a block of level l has dimensions $(2^{\lfloor \frac{l+1}{2} \rfloor} \times 2^{\lfloor \frac{l}{2} \rfloor})$.

After determining the best match for X^l, $\hat{X}^l$, the algorithm then segments the original block into two blocks, X_0^{l-1} and X_1^{l-1}, with half the pixels of the original block, and searches the dictionary of level $(l-1)$ for the elements $S_{i_0}^{l-1}$ and $S_{i_1}^{l-1}$ that minimise the cost functions for each of the sub-blocks.

After evaluating the rate-distortion (RD) results of each of the previous steps, the algorithm decides whether to segment the original block or not. Each non-segmented block is approximated by one word of the dictionary (S_i^l). If a block is segmented, then the same procedure applied to the original block is recursively applied to each segment. An example of this process is represented in figure 1, for an original block of scale 4 (4×4 pixels).

The resulting binary segmentation tree is encoded using two binary flags: flag '0' represents the tree nodes, or block segmentations and flag '1' represents the tree leafs (sub-blocks that are not segmented). These flags are not used for blocks of level 0, that can't be further segmented.

The binary tree is encoded using a preorder approach: for each node, the sub-tree that corresponds to the left branch is first encoded, followed by the right branch sub-tree. In the final bit-stream, each leaf flag is followed by an index, that identifies the word of the dictionary that should be used to

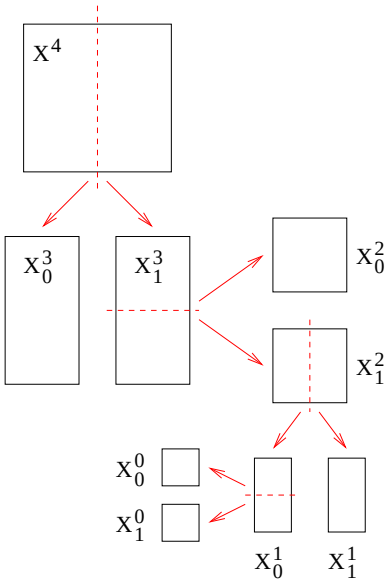

Figure 1 - Representation of the MMP algorithm coding decisions for an original 4×4 block (level 4).

approximate the corresponding sub-block. These items are encoded using an adaptive arithmetic encoder.

Figure 2 represents the coding of the example block used in figure 1 and its corresponding segmentation tree. In this example, $i_0 \ldots i_4$ are the indexes that were chosen to encode each of the sub-blocks, and so this block would be encoded using the following string of symbols:

$$0 \quad 1 \quad i_0 \quad 0 \quad 1 \quad i_1 \quad 0 \quad 0 \quad i_2 \quad i_3 \quad 1 \quad i_4.$$

The RD optimisation of the segmentation tree, $\mathcal{T}$, that is used to encode each block, is performed by evaluating the Lagrangian cost for every segmentation decision, given by

$$J(\mathcal{T}) = D(\mathcal{T}) + \lambda R(\mathcal{T}), \tag{1}$$

where $D(\mathcal{T})$ is the distortion obtained when using $\mathcal{T}$ and $R(\mathcal{T})$ is the corresponding rate.

A. Approximate block matching with scales

Unlike conventional VQ algorithms, MMP uses *approximate block matching with scales*. This block matching approach is an extension of the ordinary approximate pattern matching, in the sense that it allows the matching of vectors of different lengths.

In order to do this, MMP uses a scale transformation T_N^M to adjust the vectors' sizes, before trying to match them. For example, if we want to approximate an original block X^l, of level l and size $m^l \times n^l$, using one block S^k of the dictionary, with size $m^k \times n^k$, MMP first determines $S^l = T_k^l[S]$, a scaled version of S that has the same size of X. In other words, S^l is a version of S at level (or *scale*) l. Detailed information about the used transformations and their application in MMP is presented in [1].

In [1] the authors present a comparative study on the performance of VQ methods, whose dictionary is built from blocks of the input signal, compared to that of VQ methods whose dictionary is built using scaled versions of input blocks. The presented results give the indication that the use of dictionaries with scales can be advantageous.

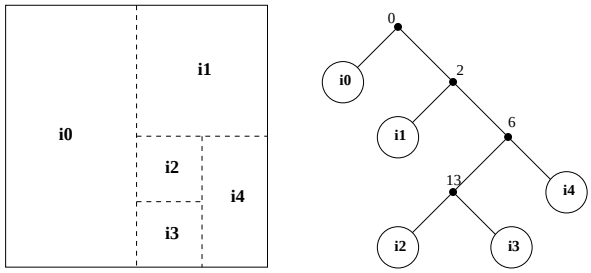

Figure 2 - Segmentation of a block and corresponding binary tree: the root corresponds to the original 4×4 block (level 4), while nodes i_2 and i_3 (1×1 blocks) belong to level 0.

B. The adaptive dictionary

MMP uses an adaptive dictionary that is updated while the data is encoded, with blocks that were used to approximate the original image data. As it has been seen, each input block of image data, X, is parsed into a set of non-overlapping segments, which are approximated by blocks of the dictionary that were scaled to the dimension of each of them.

When a given data block, X^l, is segmented, instead of being approximated by a single word of the dictionary, this block is approximated by the concatenation of the words used to approximate each of its halves. Returning once again to the example of figure 2, the block of level 1, that corresponds to node 13 of the binary tree, was segmented. This means that each of its halves will be encoded separately, using indexes i_2 and i_3 of level 0, respectively. After MMP encodes these two blocks, a new block $S^1_{i_2:i_3}$, resulting from the concatenation of the blocks $S^0_{i_2}$ and $S^0_{i_3}$ is used to update the dictionary.

Every time a block is approximated by the concatenation of two dictionary blocks, of any given level, the resulting block is used to update the dictionary, becoming available to encode the next blocks of the image, independently of their size.

This updating procedure for the dictionary uses only information that can be inferred exclusively from the encoded segmentation flags and dictionary indexes. This means that the decoder is able to keep an exact copy of the dictionary used by the encoder, using no further information.

III. The MMP Algorithm as a Signal Decomposition

From the previous presentation we can observe that, for each original input block, X, MMP determines an approximation $\hat{X}$ that is the concatenation of a set of L segments, $S_i^{l_i}$, of scales l_i, $i = 0, .., L-1$:

$$\hat{X} = [S_0^{l_0} \quad S_1^{l_1} \quad S_2^{l_2} \quad \ldots \quad S_{L-1}^{l_{L-1}}]. \tag{2}$$

Each segment $S_k^{l_k}$ is either one simple pulse, if it corresponds to one of the vectors that were used to build the initial dictionary, or a more elaborate pattern that was created by the dictionary update procedure. As was explained in the previous sections, the dictionary update procedure builds the new patterns using the concatenation of scaled versions of blocks that were already available. The nature of the dictionary update procedure means that each segment $S_k^{l_k}$ can

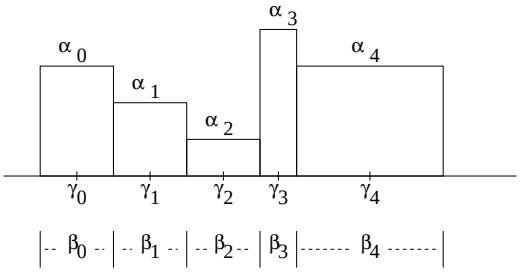

Figure 3 - Decomposition of the decoded signal into a combination of pulses with amplitudes α_k, scale β_k and position γ_k.

ultimately be represented by a combination of basic pulses, characterised by their amplitude, width and position.

This means that a decoded signal[1], $\mathcal{X}(t)$, built from the concatenation of all decoded blocks $\hat{X}$, can also be represented in terms of a combination of these modulated, scaled and shifted pulses:

$$\mathcal{X}(t) = \sum_{k=0}^{K_S} \alpha_k . \theta(\frac{t}{\beta_k} + \gamma_k), \tag{3}$$

where α_k corresponds to the amplitude of each pulse, β_k corresponds to its support, that is directly related to the MMP scale and γ_k represents each pulse position. This decomposition is represented in figure 3.

Considering the decomposition represented by equation 3, MMP can be regarded as an algorithm that determines a set of triplets, ξ_k,

$$\xi_k = (\alpha_k, \beta_k, \gamma_k), \tag{4}$$

that represent the K_S versions of the basis function θ as defined in equation 3 and efficiently encodes them. In fact, each dictionary word, S_i^l can be regarded as a particular combination of a limited number of triplets,

$$S_i^l = \{\xi_{i0}, \xi_{i1}, .., \xi_{ik_i}\}, \tag{5}$$

which means that the arithmetic encoding of one dictionary index corresponds, in fact, to a joint encoding of the set of triplets that correspond to the dictionary pattern.

IV. Problem: the use of different basis with the MMP algorithm

In the original MMP, the used basis are rectangular pulses, with means that the basis functions θ, represented in equation 3, are the scaling functions of the Haar's wavelet.

In terms of their use with MMP, these functions have, among others, the important property of creating orthogonal vectors, because they have non intersecting supports. This means that the approximation of each block can be made independently, i.e., the determined approximation for a particular block will not be affected by the influence of an adjacent vector (basis function), that will be used in the future to encode a neighbour block. This is important, because it allows for a computationally efficient optimisation of the segmentation tree used by the MMP encoder, that tries to minimise the associated cost function, given by equation 1

One first and interesting problem would be the study of other basis functions, witch would still be computationally efficient in the MMP encoder framework.

One other feature of the current MMP method is that the functions used for the analysis and synthesis are the same, meaning that the MMP decoder uses the same shifted pulses to reconstruct the signal, that were used at the encoder to optimise the signal decomposition. This fact has the disadvantage of originating some blocking artefacts on the decoded MMP image.

Notice that, at the decoder side, the computation restrictions described for the encoder do not apply. The decoder simply has to combine the vectors defined in the bit stream, in order to compose the approximated image. In fact, some tests were performed using the usual encoder but with a decoder that replaced the pulses by different synthesis functions, namely triangular and Gaussian functions, with overlapping supports [1]. The experimental results of these tests showed that these functions allowed for a reduction of the blocking artefacts in the decoded signal, but also, in the case of the Gaussians, for some increase in the objective quality, particularly for smooth images.

These interesting results raise some new questions about the use of different functions for the encoding (analysis) and decoding (synthesis) processes, namely what would be the important characteristics of these basis functions that could: first, allow for their use and second, achieve compression performance gains for MMP?

[1] We consider from now on the case of an unidimensional time signal, for the sake of easy graphical representation.

References

[1] M. de Carvalho, E. da Silva, and W. Finamore, "Multidimensional signal compression using multiscale recurrent patterns", *Elsevier Signal Processing*, , no. 82, pp. 1559–1580, November 2002.

[2] Jacob Ziv and Abraham Lempel, "Compression of individual sequences via variable-rate coding", *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[3] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*, Springer, 1991.

[4] E. B. L. Filho, E. A. B. da Silva, M. B. de Carvalho, W. S. S. Júnior, and J. Koiller, "Electrocardiographic signal compression using multiscale recurrent patterns", *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 12, pp. 2739–2753, December 2005.

[5] M. H. V. Duarte, M. B. de Carvalho, E. A. B. da Silva, C. L. Pagliari, and G. V. Mendonça, "Multiscale recurrent patterns applied to stereo image coding", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 15, November 2005.

[6] N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria, and Vitor M. M. Silva, "Universal image coding using multiscale recurrent patterns and prediction", *IEEE International Conference on Image Processing*, September 2005.

[7] N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria, and Vitor M. M. Silva, "H.264/AVC based video coding using multiscale recurrent patterns: First results", *Lecture Notes on Computer Science - VLBV 05*, 2006.