# H.264/AVC BASED VIDEO CODING USING MULTISCALE RECURRENT PATTERNS: FIRST RESULTS

*Nuno M. M. Rodrigues*[1,2], *Eduardo A. B. da Silva*[3], *Murilo B. de Carvalho*[4], *Sérgio M. M. de Faria*[1,2], *Vítor M. M. da Silva*[1,5]

[1]Instituto de Telecomunicações, Portugal; [2]ESTG, Instituto Politécnico Leiria, Portugal; [3]PEE/COPPE/DEL/Poli, Univ. Fed. Rio de Janeiro, Brazil; [4]TET/CTC, Univ. Fed. Fluminense, Brazil; [5]DEEC, Universidade 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) algorithm has been proposed recently as a universal data coding method. In addition to its efficiency for other types of signals, the MMP has proved to be a very powerful image coding method. Experimental tests showed that MMP is able to achieve better results than the traditional transform-based image coding methods, particularly for images that do not have a low-pass nature.

These promising results motivated the use of MMP for residual error encoding in hybrid video coding algorithms. This paper presents the first results of these experiments. The video encoding algorithm used in this work is based on the H.264/AVC video encoder, but uses MMP to encode the inter residual data, for the P and B slices.

Experimental results show that, even in this preliminary, not fully optimised version, this method is able to achieve an approximately equivalent performance to the H.264/AVC. This demonstrates that MMP is an alternative to the transform-quantisation paradigm for hybrid video coding that is certainly worth investigating.

## I. INTRODUCTION

Hybrid video coding schemes have been almost ubiquitous in video coding standards. They mostly use block based motion estimation and compensation to reduce the energy of the residual image. The error image is then encoded using transform coding and quantisation. Such residual error encoding is a legacy from the top image encoding methods, which traditionally use algorithms based on the transform-quantisation paradigm with excellent results.

The most recent standard for video coding, H.264/AVC (H.264) [1] also uses a transform based residual encoding method, that has been highly optimised for coding efficiency.

In this work we introduce an algorithm to encode the motion predicted data in an H.264 based video coder, which is based on an alternative paradigm to the transform-quantisation. This algorithm is referred to as Multidimensional Multiscale Parser (MMP) [2], because it uses an adaptive dictionary to approximate variable-length input vectors. These vectors result from recursively parsing an original input block of the image. Scaling transformations are used to resize each dictionary element to the dimension of the block segment that is being considered.

Previous results [2] show that MMP performs well for a wide variety of input images, ranging from smooth grayscale images to text and graphics. This lends it a universal flavour. Therefore, one expects that it should also perform well for encoding residual images. This was confirmed by results in [3], where MMP is used to encode intra prediction residuals. This motivated the use of MMP for encoding the motion compensated residual data in the H.264 video coder, replacing the adaptive block size transform (ABS) defined in this standard. We refer to it as the MMP video encoder.

This paper presents the first results of this encoder. MMP is used to encode the motion estimated residual image in P and B slices. Since our aim is to assess the performance of MMP for motion compensated residual data, the intra macroblock (MB) residues are encoded using the original H.264 transform. All other syntax elements are also encoded using the techniques defined in JM9.3 reference software [4].

Results of the first tests have shown that the MMP video encoder has an overall performance comparable to that of H.264. Taking into consideration that in these tests the rate-distortion decisions are the same ones used in JM9.3, we believe they have not been optimised for the use of MMP. This suggests that there might be some room for improvement, and therefore that it is worth investigating MMP as an alternative to the present transform-quantisation paradigm in video coding.

In the next section we briefly present the MMP algorithm for image coding. Section III describes the new MMP video encoder, and in section IV the first experimental results are presented and compared with the ones of H.264 (JM9.3, high profile). Conclusions of this work can be found in section V along with a brief outline of planned future work.

## II. THE MMP ALGORITHM

Although the MMP algorithm was initially proposed as a generic lossy data compression method, it is easily expandable to work with $n$-dimensional data, and has been successfully applied to two dimensional data. In

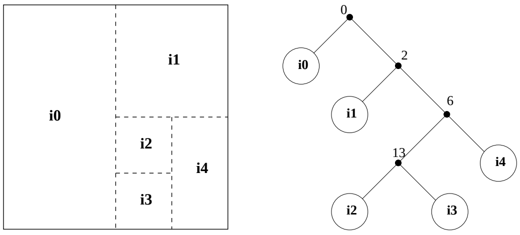

Figure 1 - Segmentation of a 4×4 block and the corresponding binary tree.

this section we describe the most important aspects of the MMP algorithm applied to image coding. An exhaustive description of the method can be found in [2].

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 (with dimensions $(2^{\lfloor\frac{l+1}{2}\rfloor} \times 2^{\lfloor\frac{l}{2}\rfloor})$). Square blocks, corresponding to even levels, are segmented into two vertical rectangles.

The algorithm then segments the original block into two blocks, $X_1^{l-1}$ and $X_2^{l-1}$, with half the pixels of the original block, and searches the dictionary of level $(l-1)$ for the elements $S_{i_1}^{l-1}$ and $S_{i_2}^{l-1}$ that minimise the cost functions for each of the sub-blocks.

After evaluating the R-D 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.

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

Figure 1 represents the segmentation of an example block and the segmentation tree that MMP uses to encode it. 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 R-D optimisation of the segmentation tree, $\mathcal{T}$, that is used to encode each block, is performed evaluating the Lagrangian cost for every segmentation decision, given by $J(\mathcal{T}) = D(\mathcal{T}) + \lambda R(\mathcal{T})$, where $D(\mathcal{T})$ is the distortion obtained when using $\mathcal{T}$ and $R(\mathcal{T})$ is the corresponding rate.

Unlike conventional vector quantisation (VQ) algorithms, MMP uses *approximate block matching with scales* and an adaptive dictionary.

Block matching with scales is an extension of the ordinary pattern matching, in the sense that it allows the matching of vectors of different lengths. In order to do this, MMP uses a separable scale transformation $T_N^M$ to adjust the vectors' size before trying to match them. For example, in order to approximate an original block $X^l$ using one block $S^k$ of the dictionary, MMP has to first determine $S^l = T_k^l[S]$. Detailed information about the use of scale transformations in MMP is presented in [2].

MMP uses an adaptive dictionary that is updated while the data is encoded. 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 future blocks of the image, independently of their size. This updating procedure for the dictionary uses only information that can be inferred by the decoder exclusively from the encoded segmentation flags and dictionary indexes. Thus, MMP has the ability to learn the patterns previously occurred on the image, adapting itself to the data being encoded. This characteristic gives it a universal flavour.

## III. THE MMP BASED VIDEO ENCODER

In this section we describe the main features of the MMP video encoder. It is based on the JM9.3 reference software of H.264 video coding standard [4]. All the encoding modes are inherited from JM9.3, as well as the rate-distortion (RD) encoding decisions.

The main difference between the MMP video encoder and H.264 is related to the motion compensated residue data encoding method for the P and B macroblocks: MMP replaces the original DCT integer transform defined in [1].

One important feature of H.264 is the use of adaptable block size transforms. Such transforms provide a significant gain in coding efficiency when compared with the fixed size blocks (either 4×4 or 8×8) used by its predecessors. Such scale adaptability allows saving bits by the use of large blocks where the residual is mostly uniform, while providing good coding accuracy through the use of small blocks in the cases where the residue data is more detailed. It is important to note that scale adaptability is a feature inherent to MMP. In fact, this is one of the strong reasons for its good coding performance.

### *A. Intra encoding*

The H.264 recommendation defines three different partition block sizes for intra MB: 16×16, 8×8 and 4×4. Intra prediction is done using four possible prediction modes for $Intra\_16 \times 16$ macroblocks and nine prediction modes for the other two partition sizes.

Previous tests compared the efficiency of MMP and H.264 in encoding the intra residue for still digital image coding. The MMP-Intra method uses a set of prediction schemes similar to those defined for H.264 intra encoding, but encodes the prediction residue with MMP. Details about this method and experimental results comparing its performance against H.264 and JPEG2000 are presented in [3].

Nevertheless, as stated in the Introduction, in this version of the MMP video encoder all the intra MB are encoded using exactly the same procedure has H.264, including the same integer DCT transform. The reason for this is that we are mainly interested in assessing the performance of MMP for inter residual coding. We do so by comparing the coding efficiency of MMP video with the one of the JM9.3 encoder. This comparison is only effective if the reference frames used by both methods is the same. This would not be the case if MMP was used for intra MB coding.

### *B. Inter encoding*

Inter MB coding involves two major steps: motion estimation and coding of the motion compensated residue. Motion estimation consists in the search of the motion vector that allows the best result for the block residue, in a RD sense. H264/AVC uses quarter-pixel precision for the motion vectors and the motion estimation is performed in three separate steps. First a full search determines the best motion vector with full pixel precision. After this, the best motion vector with half and quarter pixel precision is determined around the position obtained in the previous step.

In addition, motion compensation in H.264 is done using one of seven modes, that are related to the partitioning possibilities of a 16x16 luma macroblock. Each partition of a MB has its own motion vector, that is used in the motion compensation of the corresponding subblock. Motion estimation in H.264 thus implies the optimisation not only of the motion vectors, but also of the partition block size that is used for motion compensation.

The existence of several partition modes allows the motion compensation to be performed using a block size that optimises the distortion of the motion predicted decoded residue versus the bit-rate coding cost, corresponding to the motion compensation data plus the residue transform coefficients. In JM9.3, the cost of each mode is estimated by evaluating the distortion of the transform coding of the residues, either by using the sum of absolute differences (SAD) or the sum of absolute transformed differences (SATD).

The current version of the MMP video encoder uses exactly the same motion estimation procedure. This has implications on the efficiency of the MMP encoder, because the motion estimation process returns a set of motion vectors that are the best in the “DCT point of view”, meaning that these vectors minimise a cost function where the distortion is determined using the SA(T)D and the rate takes into account the cost of the DCT coefficients. These measurements are not related to those produced by the MMP coding, and we can expect a performance loss due to this fact. Nevertheless, this process favours a direct comparison between the encoding efficiency of MMP and the DCT, because the residue patterns that are generated by the motion estimation/compensation process tend to be approximately the same for the two encoders. One should bear in mind, though, that we can expect an improvement in performance once the motion estimation process uses a cost function more related to the MMP characteristics. We are currently working on this matter.

H.264 encodes the inter residue with a block size for the transform coding that depends on the partition size that was used for the motion estimation. The MMP video coder does not take this partition size into account and considers the entire 16×16 residue block, that is composed by the concatenation of the several partition blocks. This is not a problem because MMP is able to segment the original 16×16 block in a way that optimises the distortion of the decoded residue. Once again, the adaptability of the MMP encoding algorithm plays an important role in the efficiency of the method.

All motion compensation information, like the partition modes and the motion vectors for each block, is transmitted by the MMP video encoder using the same techniques as H.264, as described in [1].

## IV. EXPERIMENTAL RESULTS

The MMP video encoder described in the previous section was implemented and experimental tests were performed.

In this test, MMP video coder uses six independent dictionaries: one for each of the YUV components of the P and B slices. The use of different dictionaries allows each of them to efficiently adapt to the specific prediction patterns of each type of source data. This has the additional advantage of limiting the size of each dictionary, reducing the computational complexity.

When used to encode prediction error blocks, MMP uses initial dictionaries in the scale 1×1 (level 0) with values in the range from -255 to 255. The initial dictionaries for the other levels are obtained from this one by scale transformation. The scale transformation and dictionary updating procedure are the same as those described in [2].

The MMP coder was compared with the H.264 high profile video coder using version 9.3 of the reference software. Both encoders were tested using the first 99

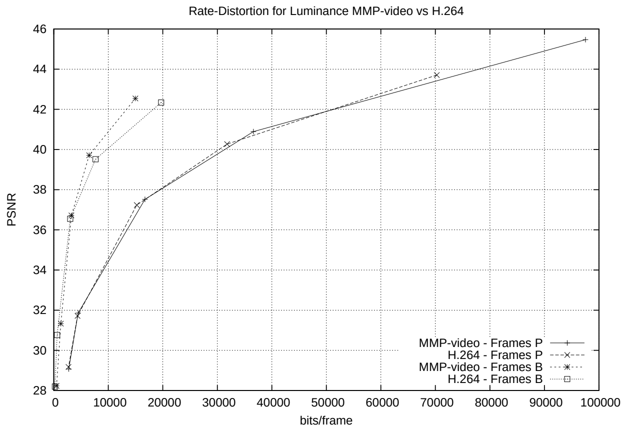

Figure 2 - Average luminance PSNR versus average number of bits per frame for P and B frames.

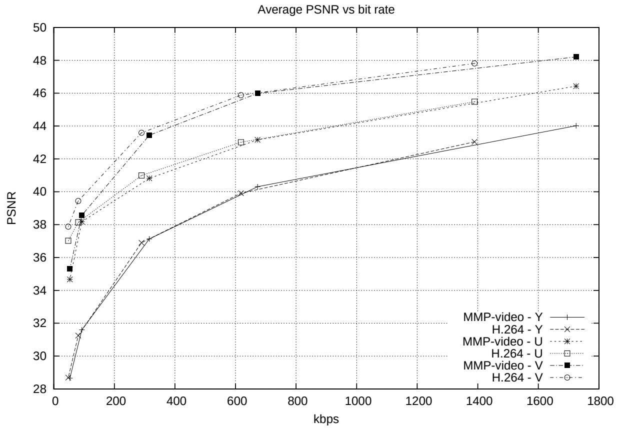

Figure 3 - Average PSNR versus bit rate for the YUV frame components.

frames of the CIF Foreman sequence, 4:2:0. Only one I frame was used with one skipped frame and one B frame (I B P B P pattern). We used the variable bit rate mode, testing the encoders for several quality levels of the reconstructed video sequence. This was done by varying the QP parameter for the I/P and B slices.

Figure 2 represents the average PSNR for the luminance component of P and B frames versus the average number of bits used to encode each frame. For the P frames the RD curves are very close, indicating that, even using sub-optimal rate-distortion decisions, MMP can perform as well as the ABS transform. Figure 2 also shows that the MMP video coder tends to achieve better RD results for coding B frames than H.264. This can be explained by the fact that B frames use bidirectional motion estimation, which generates low energy residue patterns. The H.264 encoder also uses a coarser quantisation scheme for these patterns. On the other hand, the MMP is able to “learn” the coarser residue patterns very efficiently and use them along the sequence, tending to be a better encoder for such residue blocks.

Figure 3 plots the average RD curves for each of the colour components. We can see that for luminance component, both encoders have a similar performance, but there is some loss for the chroma components for the MMP video coder. An explanation for this is the fact that, since the chroma components are much smaller and have much less energy than the luminance, then the chroma dictionaries are not able to adapt to the residue patterns, specially at low rates. However, for high bit rates, MMP segments the block and compensates the small-sized dictionaries.

A possible way to overcome this problem would be the use a single dictionary for both chroma components, or even to devise ways of making the dictionaries to grow faster (for example, by introducing extra blocks related to the original pattern).

## V. CONCLUSIONS AND FUTURE WORK

This paper presents the first results of the use of the Multidimensional Multiscale Parser (MMP) for hybrid video coding. MMP is used instead of the integer DCT in a H.264/AVC based video coder, to encode the prediction residual data of the motion compensated macroblocks.

Experimental results have shown that the MMP video coder is able to achieve better results than H.264 for the luminance component of B and P slices, but still has some losses for the chroma components. Nevertheless, the current results demonstrate that MMP is an alternative to the integer DCT used by H.264 that is worth investigating.

The results presented in this paper show much room for further improvements, because the MMP video encoder has not yet been thoroughly optimised in a rate-distortion sense. Future work will address this issue, as well as several other questions that are relevant to the performance of the MMP video coder. Among them, one can distinguish the use of adaptive block size MMP to encode the motion compensated residue partition blocks, the use of MMP on intra MB and the optimisation of the deblocking filter to the MMP reconstructed frames.

## References

[1] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6), *Draft of Version 4 of H.264/AVC (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding),* March 2005.

[2] 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.

[3] Nuno M. M. Rodrigues, Eduardo A. B. da Silva, Murilo B. de Carvalho, Sérgio M. M. Faria, and Vítor M. M. Silva, “Universal image coding using multiscale recurrent patterns and prediction,” *IEEE International Conference on Image Processing, Genova, Italy,* September 2005.

[4] http://iphome.hhi.de/suehring/tml/download/.