An Efficient H.264-based Video Encoder Using Multiscale Recurrent Patterns

Nuno M. M. Rodrigues[ab], Eduardo A. B. da Silva[c], Murilo B. de Carvalho[d], Sérgio M. M. Faria[ab] and Vitor M. M. da Silva[ae]

[a] Instituto de Telecomunicações, Universidade de Coimbra - Pólo II, Coimbra, Portugal
[b] Escola Superior Tecnologia e Gestão, Instituto Politécnico de Leiria, Portugal
[c] Programa de Engenharia Elétrica - COPPE/UFRJ, Rio de Janeiro, Brazil
[d] Universidade Federal Fluminense, Niterói, Brazil
[e] Dep. Eng. Electrotécnica e de Computadores, Universidade de Coimbra - Pólo II, Portugal

ABSTRACT

We investigate new and efficient methods for coding the motion compensated residues in a hybrid video coder framework, that are able to improve upon the performance of the very successful DCT based adaptive block size integer transform used in H.264/AVC.

We use an algorithm based on adaptive block size recurrent pattern matching, that encodes each block of motion compensated predicted data using a scaled pattern stored into an adaptive dictionary. We refer to this algorithm as Multidimensional Multiscale Parser, or MMP.

A video encoding method is presented, that uses MMP instead of the integer transform in an H.264/AVC encoder framework. Experimental results show that MMP is capable of achieving consistent gains on the final average PSNR, when the new encoder is compared with H.264/AVC's *high* profile.

Keywords: Video coding, motion compensated residue coding, approximate pattern matching

1. INTRODUCTION

In spite of the ever growing available bandwith and network speeds, the need for efficient video encoding algorithms is as important now as it has ever been. This fact led to the combination of efforts from the two main video coding standards developing groups: the ITU-T's Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). One product of this joint effort was the H.264/AVC video coding standard,[1] that represents the state-of-the-art of video encoders.

The H.264/AVC standard defines a set of new coding tools, that improve its efficiency to about two times that of previous standards, at the cost of an increased computational complexity. In spite of the importance of these new coding tools, H.264/AVC maintains the traditional hybrid coding architecture and also uses a transform-quantisation based encoding scheme, to compress the motion compensated residue patterns.

In this paper we present a video encoder that uses a new paradigm to encode the motion compensated residue data, based on approximate pattern matching. The proposed encoder replaces the traditional DCT-based transform-quantization encoder by a recently proposed encoding method, named Multidimensional Multiscale Parser, or MMP.[2]

MMP is a generic data encoding algorithm, that uses *pattern matching with scales* to encode each original image block with a vector from an *adaptive dictionary.* As a generic encoding algorithm, MMP is highly adaptable to a wide set of different data sources, and some previous works demonstrated its good performance for encoding a wide variety of data types, that range from electrocardiography data (ECG) to image and stereoscopic image coding.[3–5]

Further author information:
E-mails: nuno.rodrigues@co.it.pt, eduardo@lps.ufrj.br, murilo@telecom.uff.br, sergio.faria@co.it.pt, vitor.silva@co.it.pt

In the next sections we start by briefly introducing the state of the art of the class of MMP algorithms for image encoding. After this we describe the new video encoder, that we refer to as MMP-Video. Several aspects of this encoder will be discussed and commented on. Finally, we present some experimental results for the discussed methods, that are compared among them and with the results for the H.264/AVC *high* profile video encoder and we draw some conclusions from this work.

2. MMP FOR IMAGE ENCODING

MMP encodes each original image block by approximating it with a vector from an *adaptive dictionary* $\mathcal{D}$. This is done by using *different scales*, meaning that blocks of different dimensions can be approximated by this procedure. These dimensions correspond to successive binary segmentations of an original square block, first in the vertical, then in the horizontal direction. MMP can be summarised by the following main steps:

For each block of the original image, $\mathbf{X}^{l}$*:

1. find the dictionary element $\mathbf{S}_i^l$ that minimises the Lagrangian cost function of the approximation, given by: $J(\mathcal{T}) = D(\mathbf{X}^l, \mathbf{S}_i^l) + \lambda R(\mathbf{S}_i^l)$, where $D(.)$ is the sum of square differences (SSD) function and $R(.)$ is the rate needed to encode the approximation;
2. parse the original block into two blocks, $\mathbf{X}_1^{l-1}$ and $\mathbf{X}_2^{l-1}$, with half the pixels of the original block;
3. apply the algorithm recursively to $\mathbf{X}_1^{l-1}$ and $\mathbf{X}_2^{l-1}$, until level 0 is reached;
4. based on the values of the cost functions determined in the previous steps, decide whether to segment the original block or not;
5. if the block should not be segmented, use vector $\mathbf{S}_i^l$ of the dictionary to approximate $\mathbf{X}^l$;
6. else
   (a) create a new vector $\mathbf{S}_{new}^l$ from the *concatenation* of the vectors used to approximate each half of the original block: $\mathbf{X}_1^{l-1}$ and $\mathbf{X}_2^{l-1}$;
   (b) use $\mathbf{S}_{new}^l$ to approximate $\mathbf{X}^l$;
   (c) use $\mathbf{S}_{new}^l$ to *update* the dictionary, making it available to encode future blocks of the image.

When applied recursively, this algorithm generates a binary segmentation tree for each original block, that is encoded using two binary flags ('0' for the tree nodes, or block segmentations and '1' for tree leafs, or unsegmented blocks). This binary tree is encoded using a top-bottom preorder approach. In the final bit-stream, each leaf flag is followed by an index, that identifies the vector of the dictionary that should be used to approximate the corresponding sub-block. These items are encoded using an adaptive arithmetic encoder.

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

Block matching with scales allows for 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' sizes before attempting to match them. For example, in order to approximate an original block $\mathbf{X}^l$ using one block $\mathbf{S}^k$ of a different scale of the dictionary, MMP first determines $\mathbf{S}^l = T_k^l[\mathbf{S}]$. Detailed information about the use of scale transformations in MMP is presented in.[2]

The use of an adaptive dictionary is illustrated by the final step of the previous algorithm. Every segmentation of a block from level l originates the concatenation of two dictionary blocks of level $l-1$. 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, since it is based exclusively in the encoded segmentation flags and dictionary indexes.

*The superscript l means that the block $\mathbf{X}^l$ belongs to *scale* l or *level* l of the segmentation tree (with dimensions $(2^{\lfloor\frac{l+1}{2}\rfloor} \times 2^{\lfloor\frac{l}{2}\rfloor})$).

2.1. Efficient Dictionary design

MMP uses an initial dictionary consisting of a few blocks with constant value. This highly sparse initial dictionary is obviously not efficient for coding images, but the updating procedure quickly adapts the patterns in the dictionary to the typical patterns that are being encoded.

Observations of the final dictionary sizes showed that the final number of blocks for each level of the dictionary is, by far, much larger than the total number of blocks that are actually used. This process has the disadvantage of increasing the average bit rate needed to transmit each index of the dictionary, that grows with the total number of indexes, compromising the method's performance.

In order to avoid this, a test condition was added to the dictionary update procedure, to ensure that the quadratic distortion between each new block of level l, $\mathbf{S}^l_{new}$, and the ones already available in the dictionary is not inferior to a given threshold d. This introduces a "minimum distance condition" between any two vectors of each level of the dictionary. The optimum value of d is a function of the target bit-rate and therefore of the parameter λ. This procedure eliminates the redundancy between the dictionary elements and increases the overall performance of the encoder.[4]

2.2. The MMP-Intra algorithm

The original MMP algorithm is able to achieve a very good performance for several types of images, like text, graphics and compound images, where it outperforms state-of-the-art still image coders, like JPEG2000[6] and H.264-Intra, but is not able to achieve this kind of performance for smooth images. MMP-Intra was developed in an attempt of improving the performance of MMP for smooth images. MMP-Intra is a successful combination of the original MMP algorithm with intra-frame prediction techniques.[7]

MMP-Intra uses essentially the same prediction modes used by H.264/AVC intra coded blocks to determine, for each original block, $\mathbf{X}^l$, a prediction block, $\mathbf{P}^l_m$, and the respective residue values, $\mathbf{R}^l_m$. The block with residual data is then encoded using MMP. The hierarchical prediction scheme uses adaptive block sizes and Lagrangian rate-distortion (RD) cost functions, that allow the encoder to optimise the block prediction, determining the best trade-off between the prediction accuracy and the additional overhead introduced by the prediction data.

Recent experimental results showed that MMP-Intra has a coding performance similar to that of the state-of-the-art transform based image encoders for smooth images, while maintaining its performance advantage for non-smooth images.[4]

3. MMP FOR VIDEO ENCODING

The good results achieved by MMP-Intra demonstrate that MMP is a very efficient method to encode image residue patterns that result from a prediction process, in this case, intra-frame prediction. The comparison with H.264/AVC intra-frame encoder shows that approximate pattern matching with scales can, in some cases, be advantageous over the H.264/AVC's integer DCT, when both are used to encode predicted intra-frame error.

This fact led us to investigate the use of MMP to encode the motion compensated residual data, in a hybrid video coding scheme. The resulting encoding algorithm was named MMP-Video, because it uses MMP for the compression of the motion compensated residual data, in an H.264/AVC video coding framework. H.264/AVC was chosen because of its top performance when compared with other video encoders described in literature. Initial results for MMP-Video demonstrated its good potential for video coding despite its minor quality losses when compared with H.264/AVC.[8]

Since then, work has been done in the investigation of more efficient ways to integrate the MMP encoding process in the H.264/AVC-based hybrid video coder framework. This section starts with an overall description of the MMP-Video encoder and then discusses the functional improvements and architectural adaptations. One of the factors that causes significant impact on the algorithm performance is the dictionary architecture. The studied alternatives are presented and discussed.

3.1. Basic MMP-Video architecture

As was previously referred, the main idea behind MMP-Video is the use of an MMP-based encoder instead of the integer version of the DCT originally defined in the H.264/AVC standard.[1] Because of this, the MMP-Video implementation shares the same structure of the H.264/AVC reference software (version JM9.3)[9] but uses MMP to encode the motion compensated residue data of the inter macroblocks (MBs).

As H.264/AVC, the MMP-Video encoder also uses adaptive block sizes (ABS) to perform the motion estimation/compensation of the MBs. The motion estimation process, performed at the encoder, has the objective of optimizing the partition modes and corresponding motion vectors (MVs), for each MB. This is done by minimizing the RD Lagrangian cost function, $J_{(M_i,MV_i)}$ for all combinations of partition modes, M_i, and corresponding MVs. This motion compensation data is then encoded and transmitted, followed by the corresponding motion compensated residue block. Instead of the integer ABS version of the DCT transform used by H.264/AVC, the MMP-Video encoder uses the MMP algorithm. Apart from the motion compensated residual data, all information transmitted by MMP-Video is encoded using the same techniques applied by H.264/AVC.

MMP-Video also evaluates the value of the RD cost function using the same measurements as H.264/AVC, i.e., it estimates 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). Unlike the case of the H.264/AVC's original transform, the residue block with minimal SAD or SATD is not necessarily the block that is more efficiently encoded by MMP. This is sub-optimal from the MMP coding point of view, but our simulations have demonstrated that the use of these error measures still allows MMP-Video to perform efficiently, with a significant reduction in computational complexity. This complexity reduction results from the simplification inherited from the original encoder, that replaces a whole MB's residue direct and inverse transform evaluation with a simpler SAD or SATD calculation. This has the interesting side effect of allowing for a fair comparison between the encoding efficiency of MMP and the DCT, because the residue patterns, generated by the motion compensation process, tend to be approximately the same for both encoders.

For intra MBs, MMP-Video uses the original prediction modes of H.264/AVC, as well as its DCT transform. This happens because the performance of MMP and the DCT for intra frames can be different,[4] which would compromise the comparison of the performance of MMP-Video and H.264/AVC for coding motion compensated residual errors, that would use different intra reference frames.

3.2. The use of a CBP-like flag

In H.264/AVC, one encodes, for each MB, a *Coded Block Pattern* (CBP) parameter, that characterizes the encoded residue block transform coefficients (for instance the absence of non-zero AC coefficients). This allows the encoder to avoid transmitting information associated with some null residual coefficients, saving overhead bits. Since MMP does not use encoded coefficients, the direct use of the CBP value is not possible. Nevertheless, an analysis of the MMP-Video coding data revealed that a large number of null 16×16 and 8×8 blocks were transmitted. This can be a source of inefficiency for the MMP-Video encoder. For a null residual block, MMP has to transmit one no-segmentation flag followed by the index that corresponds to the null block, for each of the three components, i.e., a total of six symbols.

Two ways of reducing this inefficiency in the MMP-Video encoder were studied. Both use the transmission of CBP-like information. In the first case, a binary flag signals the existence of non-zero residue blocks for any of the YUV components. If all component residues are null, this flag (we'll call it CBP, as a reference to its purpose in the original H.264/AVC encoder) will be zero and the encoder simply does not transmit any MMP information for the current MB, thus saving bits for the same reconstruction quality. When there are non-null residues, the CBP flag is one and its transmission is followed by the encoded MMP residue blocks.

The second version of this process uses a CBP value with three bits, that signal independently whether there are non-null encoded residue values for each of the Y, U and V components. As an example, a CBP with value 5 is followed by the MMP encoded residues for the luma and chroma-V components and explicitly signals the decoder that the chroma-U residue is null.

In both cases, the CBP value is encoded using an adaptive arithmetic encoder and transmitted for every MB immediately before the MMP data that encodes the residue patterns. This CBP value introduces an additional overhead in the MMP-Video encoder.

3.3. Dictionary design for MMP-Video

Some previous studies showed that the performance of MMP for image coding, depends on the way dictionary is built.[4,7] Our investigation on MMP-Video demonstrated that in this case, the dictionary design aspects of the encoder can be even more important.

One important factor is the use of a dictionary redundancy control scheme, already discussed in section 2.1. Because of this, MMP-Video only uses a given block to update the dictionary if the quadratic distortion between this block and all other blocks in the dictionary is below a given threshold. As for the case of image encoding, the value for the best distortion threshold depends on the target rate and is related (in the exact same way) to the value of λ.[4] The value of this parameter λ of the MMP-Video encoder is set using the same value as the quantization parameter, QP, used in the original H.264/AVC encoder.

Another important aspect of the dictionary design process for MMP-Video is the use of independent dictionaries for each image component or slice type. In its initial version, MMP-Video coder used six independent dictionaries to encode the MBs of each of the YUV components of the P and B slices. In this case, each dictionary only "learns" the specific residue patterns of each type of source data. On one hand, this can be a performance improvement factor, because it results in highly specialized dictionaries. On the other hand, this means that a MB of a given component/slice type can only be approximated by the blocks of the corresponding dictionary, that has a smaller approximation power than a more general (and thus more complete) dictionary. This fact became evident for the chroma's componets. Since chroma residue blocks tend to be very smooth, the number of segmentations performed by MMP for these blocks is small. This causes the chroma dictionaries to learn very few new patterns, which compromises their coding efficiency.

Several tests were performed in order to evaluate which was the best design for the MMP-Video's dictionary(ies). Two options were shown to be capable of achieving the best overall results:

- In the first option, MMP-Video uses two *independent* dictionaries: one for all three components of the P slices' MBs and another for all components of the B slices' MBs. This means that each dictionary is able to learn the residue patterns that correspond to all three components of every MB of each slice type, allowing MMP to use a richer dictionary to encode the chroma components, improving the coding efficiency of these residues. On the other hand, luma MBs have to "share" the dictionary with the chroma patterns, causing a slight efficiency loss for the luma component. Nevertheless, this configuration achieved the best overall results when we used independent dictionaries, because the small loss for the luma components is compensated by the relevant gains achieved for the chroma components.
- In the second option, *one single* dictionary was used to approximate all residue blocks, independently of their corresponding component and slice type. This dictionary was, however, segmented considering the original level of each new block of the dictionary, using a similar procedure to the one that has been studied for still images.[4] This process corresponds to the use of a context adaptive arithmetic encoder for every dictionary index: instead of indexing the entire dictionary, the indexes are divided into groups, or segments, using a classification criterion. Using such a procedure, each index is transmitted using one symbol specifying the dictionary segment, followed by another symbol, that identifies one index inside that segment.

The use of a segmented dictionary, like the one described in the second option, allows for a context adaptive arithmetic coding of the dictionary indexes, because different probability histograms (contexts) can be used to encode the dictionary segment symbols and indexes, in a way that may allow for a better exploitation of the statistical dependencies between the MMP symbols. In MMP-Video, the indexes are grouped according to the original scale of the block that is used to update the dictionary, i.e., segment l of the dictionary contains only those blocks that were inserted in the dictionary resulting from a new vector of scale l.

Other options were investigated, that use a larger number of independent dictionaries and different segmentation criteria (like the slice type of the MB), but attained inferior results to those achieved using the two previously described versions.

After investigating the most favorable dictionary architectures for the MMP-Video, some simulations were performed in order to determine which of the CBP parameter configurations, discussed in section 3.2, were more efficient for the two considered cases. Experimental results demonstrated that the best option for the use of a CBP parameter in MMP-Video depends on the dictionary configuration:

- when MMP-Video uses two independent dictionaries for each slice type, the best option is to encode a CBP parameter with three bits and use it for both P and B slices. In this case, the three-bit CBP is able to efficiently avoid the transmission of null residue for some of the components and was shown to be more efficient than the use of a binary valued parameter.
- when all residue blocks are encoded with only one segmented dictionary, it is more efficient to use a binary CBP and transmit this value only for the B slices' MBs. In this case, the overhead introduced by the use of a three-bit CBP results in a loss of coding efficiency, as does the use of the binary CBP for the P slices.

Since the MMP algorithm is based on a full search approximate pattern matching scheme, its computational complexity tends to be similar to that of traditional vector quantization (VQ) methods. Because of this, MMP-Video is more complex than the H.264/AVC encoder. At this stage of our investigation we are more concerned with increasing the encoding performance of the proposed algorithms. Future work will address some complexity reduction issues, through the use of previously proposed techniques, that were shown to be very efficient in reducing MMP image encoder's computational complexity without compromising its encoding efficiency.[4]

4. EXPERIMENTAL RESULTS

In this section we present some experimental results for the two versions of the MMP-Video encoder, discussed in the previous section, and compare them with the results of the H.264/AVC *high* profile video encoder, for the first 99 frames of the well known test sequences: Foreman (CIF), Akiyo (QCIF) and Mobile and Calendar (CIF). We tested the colored versions of these sequences, using the 4:2:0 YUV versions.

Besides the differences that result from the different residue pattern encoding algorithm, all encoders used the same encoding parameters, namely: a IBPBP pattern with only one intra reference frame, *high* profile (for H.264/AVC this also allows the use of 8x8 DCT transform block size and corresponds to the best encoding performance), RD optimization enabled, no error resilience and no weighted prediction for B frames. The context-based adaptive arithmetic coder (CABAC) option was set for all encoders. Variable bit rate mode was used and the encoders were tested for several quality levels of the reconstructed video sequence, by fixing the QP parameter for the I/P and B slices.

Figures 1 and 2 show the results for the P and B slices. These results are presented for luma (Y) and chroma (U and V) separately. The presented RD lines represent the average PSNR values versus the average number of bits per frame.

When we compare the performances of the two described versions of the MMP-Video encoder we observe that they vary in performance but have generally the same behaviour. The version that uses one common dictionary for all components and slice types (MMP-Video 1Dict) tends to be marginally better for the chroma components of the P slices. On the other hand, when MMP-Video uses two independent dictionaries, one for all components of the P slices and the other for B slices (MMP-Video 2Dicts), the attained results have some advantage for the luma components of both P and B slice types.

When compared with the performance of the H.264/AVC *high* profile video encoder, figures 1 and 2 show that the performance of the MMP-Video encoder is consistently better for all components of B slices, specially at higher bitrates. For these rates we may observe PSNR gains that range up to 2dB. For the P slices we observe that MMP-Video is able to achieve some coding gains over the H.264/AVC encoder for the chroma components, for all sequences except for Akiyo. For the luma component of the P slices, the relative performance of MMP-Video varies with the tested sequence: it is better than the H.264/AVC for the Mobile and Calendar sequence, equivalent for the Foreman sequence and slightly worse for Akiyo.

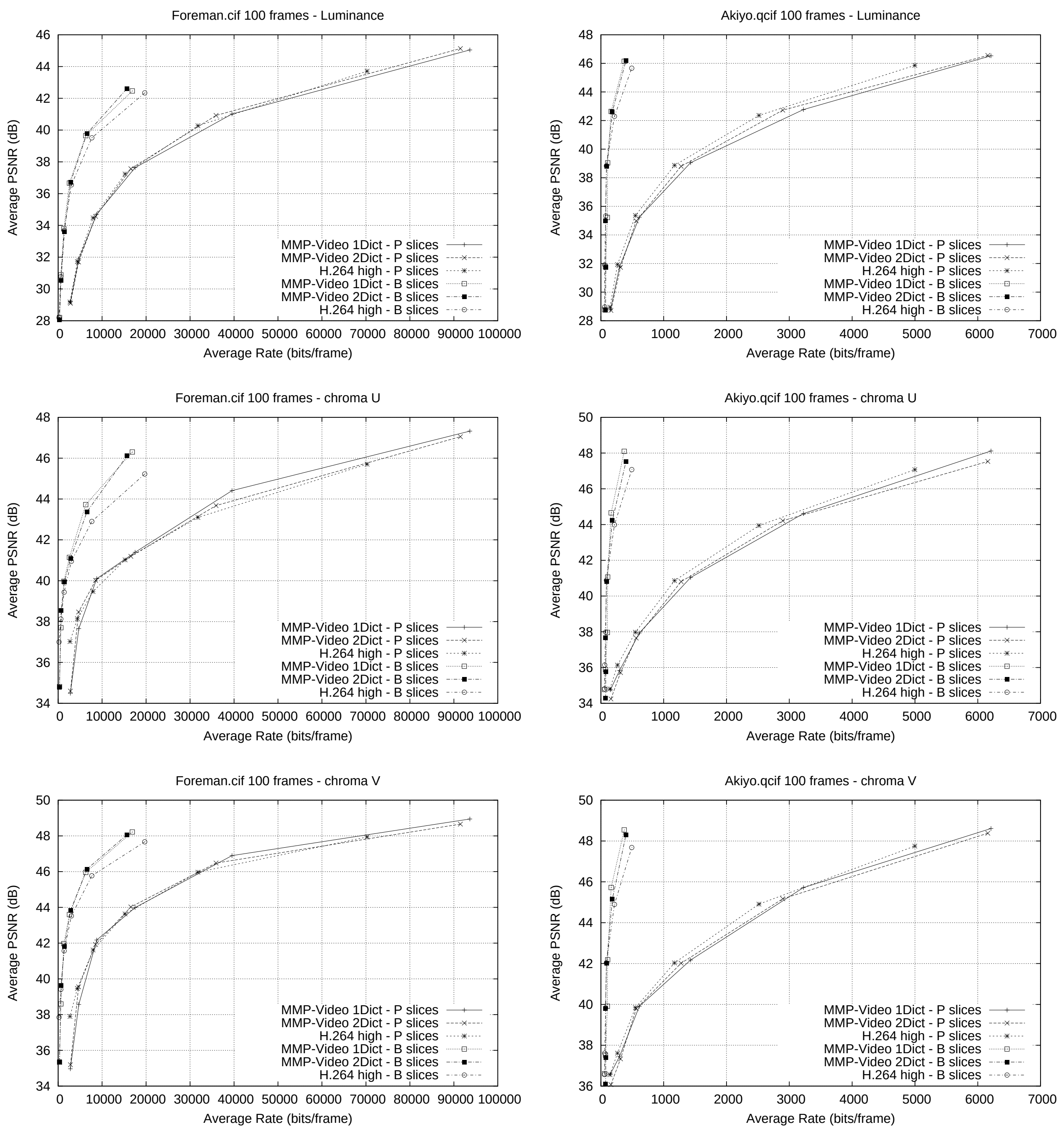

Figure 1. Sequences Foreman (CIF) and Akiyo (QCIF): comparative results for the two discussed versions of the MMP-Video encoder, that use one common dictionary for all components/slice types (MMP-Video 1Dict) or two independent dictionaries, for P slices and for B slices (MMP-Video 2Dicts) and the H.264/AVC *high* profile video encoder.

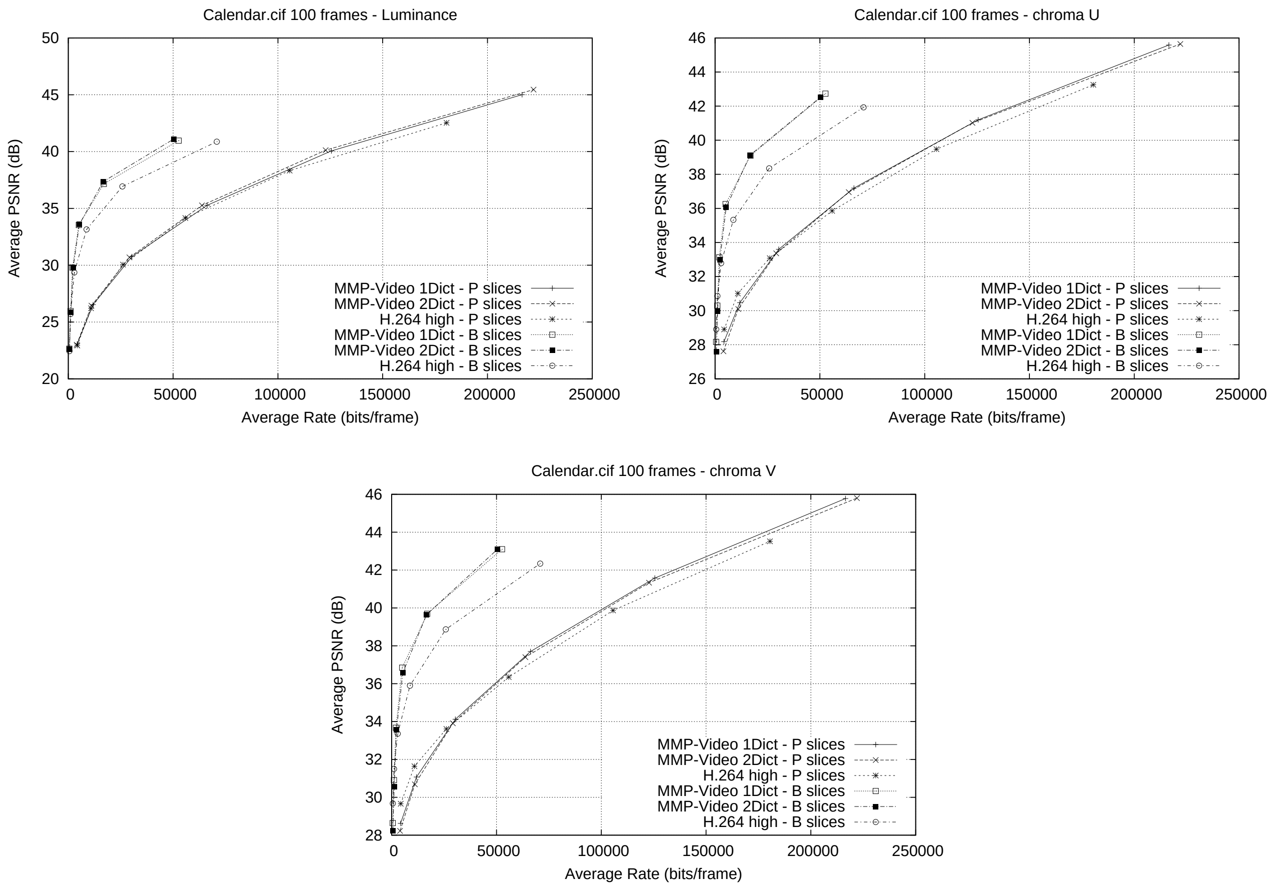

Figure 2. Sequence Mobile and Calendar (CIF): comparative results between the MMP-Video encoder and the H.264/AVC *high* profile video encoder.

5. CONCLUSIONS

We describe a new video coding paradigm that uses an hybrid architecture associated with an algorithm for coding motion compensated residue using scale adaptive approximate pattern matching, based on the Multidimensional Multiscale Parser encoding method.

The use of MMP in this hybrid video coder framework must take into consideration some important functional optimisations. These techniques, described in the text, use results derived from previous work on MMP as a digital image encoder, but also techniques that were specially designed in order to cope with the particular characteristics of digital video encoding.

Experimental results comparing the MMP-Video encoder with the H.264/AVC video encoder are presented. These results show that the general coding performance of the MMP-Video encoder is better than the one of the H.264/AVC encoder (even when the *high* profile is used), specially for medium to high bit-rates and for the B slice data, where the coding gains range up to 2dB.

The good results of MMP-Video show that, as for other previously tested data sources, the coding of motion compensated residuals takes good advantage of the adaptability of MMP. In fact, in spite of its larger computational complexity, MMP has the relevant advantage of being an encoder with universal characteristics, that is able to "learn" the typical patterns of any data source (like those of motion compensated residue blocks) and store them in the adaptive dictionary.

REFERENCES

1. J. V. T. J. of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and I.-T. S. 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* , pp. 1559–1580, November 2002.
3. 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* **52**, pp. 2739–2753, December 2005.
4. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria, V. M. M. Silva, and F. Pinagé, "Efficient dictionary design for multiscale recurrent patterns image coding," *ISCAS 2006 IEEE International Symposium on Circuits and Systems* , May 2006.
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* **11**, November 2005.
6. D. S. Taubman and M. Marcelin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, 2001.
7. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria, and V. M. M. Silva, "Universal image coding using multiscale recurrent patterns and prediction," *IEEE International Conference on Image Processing* , September 2005.
8. N. M. M. Rodrigues, E. A. B. da Silva, M. B. de Carvalho, S. M. M. de Faria, and V. M. M. Silva, "H.264/AVC based video coding using multiscale recurrent patterns: First results," *Lecture Notes on Computer Science - VLBV 05* **3893**, pp. 107–114, February 2006.
9. http://iphome.hhi.de/suehring/tml/download/.