VIDEO CODING USING PRUNED TRANSFORMS AND INTERLEAVING OF MULTIPLE BLOCKS

Cixun Zhang[*], *Kemal Ugur*[§], *Jani Lainema*[§], *Antti Hallapuro*[§], *Moncef Gabbouj*[*]

[*]Tampere University of Technology, Tampere, Finland
[§]Nokia Research Center, Tampere, Finland

ABSTRACT

Technologies used in today's video coding standards have been designed and optimized mainly for Standard Definition (SD) resolutions and below. When moving to higher resolutions, data in video frames tend to become more correlated spatially. In this paper, we study how to take advantage of this phenomenon to lower computational requirements for High Definition (HD) video coding. A coding method based on low complexity pruned transforms and interleaving of multiple transform coefficient blocks is proposed. An example implementation of this method in the context of H.264/AVC is also presented. Experimental results show that the proposed method maintains the high compression efficiency of H.264/AVC while significantly lowering the coding complexity of the codec.

Index Terms— Pruned transform, interleaving, HD video coding, H.264/AVC, complexity reduction

1. INTRODUCTION

The latest international video coding standard H.264/AVC provides up to 50% gain in coding efficiency compared to previous standards. However, this is achieved at the cost of increased encoding and decoding complexity. It is estimated that the encoder complexity increases with more than one order of magnitude between MPEG-4 Part 2 (Simple Profile) and H.264/AVC (Main Profile) and with a factor of 2 for the decoder [1]. For mobile multimedia use-cases, additional complexity of H.264/AVC becomes an issue due to the limited resources of battery operated mobile devices. On the other hand, as display resolutions and available bandwidth increases rapidly, High-Definition (HD) video is becoming more commonly used, making the implementation of video codecs even more challenging for mobile devices.

There have been recent research efforts to improve the performance of H.264/AVC for HD resolutions. In [2] Dong et al. use a large 16x16 transform for improved coding efficiency at high resolutions. In [4] it was shown that extending the macroblock size to larger than 16x16 will improve the coding efficiency for HD video especially at lower bit-rates because bits for coding side information is reduced. However, it is noted that these coding tools bring coding efficiency with increased complexity. For example, the 16x16 transform proposed in [2] is approximately four times more complex than the 4x4 transform in H.264/AVC [2][3]. Also, when using macroblock structures larger than 16x16 pixels, the encoder complexity will increase especially if Rate Distortion Optimized (RDO) encoding is utilized due to the large number of available coding modes for the macroblock [4].

In this paper, we propose a novel method targeted for coding HD resolution video achieving similar coding performance with H.264/AVC, but with significantly lower complexity. Proposed method is based on generating residual coefficients using pruned transform, interleaving the resulting coefficients of multiple transform blocks into one block and jointly coding them. In this algorithm, only a subset of the coefficients of a traditional full transform is coded. This way, an encoder or decoder implementing the proposed algorithm could utilize a pruned transform to reduce the complexity. The proposed algorithm is implemented to code the residual of INTRA predicted blocks in the H.264/AVC standard. It should be noted that this implementation is one of many potential use of the proposed algorithm, and it could be extended to different implementations, including the 16x16 transform and super macroblock ideas mentioned above. It is observed that proposed algorithm reduces the decoding complexity of INTRA coding significantly with negligible penalty on coding efficiency. The paper is organized as follows: The proposed algorithm is introduced in section 2 and the specific implementation of the proposed method in the context of H.264/AVC INTRA coding is described in section 3. Detailed complexity analysis and experimental results are given in section 4 and section 5, respectively. Section 6 concludes the paper.

2. JOINT CODING OF MULTIPLE BLOCKS

Typical hybrid video codecs code the video signal or the prediction error block by block, by transforming it using a specified transform (e.g. Discrete Cosine Transform (DCT) or a variant of it), quantizing the coefficients and entropy coding the quantized coefficients. This traditional approach

of coding the coefficients of each transform block independently is not efficient in certain cases because:

- Correlation between neighboring pixel, block or macroblock is higher when the resolution increases. Coding some blocks together helps to exploit this correlation further.
- For high-resolution video signal, a block covers a relatively small area compared to a lower resolution signal and does not contain many details. This means that transform coefficients of a block typically has low energy at its high frequency components. Therefore, we can safely discard the high-frequency coefficients to decrease the bit rate and keep the coding efficiency performance.
- Side information related to entropy coding of coefficients, such as indication of end of block etc. is coded multiple times for each block. In certain cases, this increases the encoding and decoding complexity of entropy coder and also hurts the coding efficiency.

All the above points result in unnecessary increase in bitrate required to represent coded signal with also an increase in complexity. Based on this observation, we propose a novel method for coding HD resolution video signal achieving similar coding performance with H.264/AVC, but with significantly lower complexity. Proposed method is based on generating residual coefficients using pruned transform, interleaving the resulting coefficients of multiple transform blocks into one block and jointly coding them. This way, an encoder or decoder implementing the proposed algorithm could utilize a pruned transform to achieve lower complexity. As will be shown later, proposed method is an effective and compact way to code large stationary areas with very low complexity.

In order to demonstrate its effectiveness, the proposed algorithm is implemented in the context of the H.264/AVC video coding standard to efficiently code the residual signal of INTRA predicted blocks. More specifically, INTRA_16x16 mode of H.264/AVC is replaced by the proposed algorithm. The reason for choosing to use INTRA_16x16 mode is because it was also designed to efficiently code the flat regions, hence it is suitable to be used as a comparison. However, it should be noted that this implementation is one of many potential uses of the proposed algorithm. For example it could be extended to different implementations, such as coding residual of INTER blocks or utilizing different size blocks. The implementation details of the proposed algorithm are presented in the next section.

3. CODING INTRA RESIDUAL USING THE PROPOSED ALGORITHM

The INTRA_16x16 mode in H.264/AVC is designed to code large flat areas in a video signal in an efficient and compact way. In this mode, the prediction is first formed for the whole macroblock using the reconstructed pixels that are above and left of the macroblock. Then the residual macroblock is divided into blocks of size 4x4, and each block is transformed, quantized and entropy coded. An extra hadamard transform is applied to the DC coefficients of each coefficient block to achieve higher coding efficiency.

The modified INTRA_16x16 mode implementing the proposed algorithm is illustrated in Figure 1. In this implementation, the prediction for the macroblock is obtained in the same way as in H.264/AVC. Then the residual macroblock is divided into 4 8x8 blocks (indicated with blocks *A,B,C,D* in Figure 1) and pruned transform is applied for every 8x8 block and results in coefficients of low frequency components of each block (indicated with gray blocks, A_{sub}, B_{sub}, C_{sub}, D_{sub} in Figure 1). The resulting coefficients of the four blocks are then quantized and interleaved into a single block (block *S* in Figure 1) and coded into the bitstream. Two key steps of the proposed algorithm are: using pruned transform to obtain low frequency components of transform blocks, and the interleaving process to combine the transform coefficients from multiple blocks into a single block. The details of these two steps are given in the next subsections.

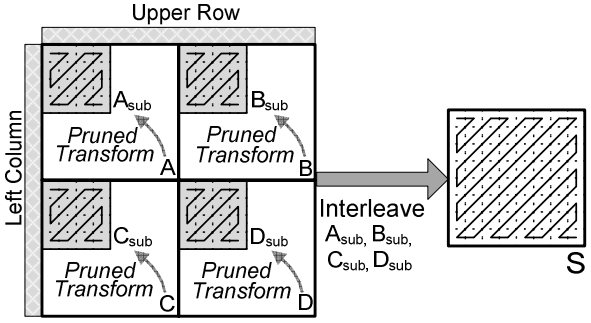

Figure 1. Illustration of Proposed Algorithm implemented to code INTRA_16x16 mode of H.264/AVC

3.1. Pruned transform

Pruned transform is used to obtain the 4x4 transform block from the 8x8 residual signal (i.e. obtaining A_{sub}, B_{sub}, C_{sub}, D_{sub} from *A*,*B*,*C*,*D* respectively). The transform size is 8x8, but only the coefficients in the upper-left 4x4 block are calculated. This structure makes the hardware and software implementations of pruned transform more regular and easier with similar coding efficiency compared to other more complex pruned transform structures that, for example, select the coefficients in zigzag scan order [5].

Assume the input data is given in the 8x8 matrix **X** and 8x8 transform matrix is given in **T**. They could be written as

$$X = \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}, \; T = \begin{bmatrix} T_{00} & T_{01} \\ T_{10} & T_{11} \end{bmatrix}$$

where $\mathbf{T_{00}}$, $\mathbf{T_{01}}$, $\mathbf{T_{10}}$, $\mathbf{T_{11}}$, $\mathbf{X_{00}}$, $\mathbf{X_{01}}$, $\mathbf{X_{10}}$, $\mathbf{X_{11}}$ are the 4x4 sub-matrices of **X** and **T** respectively. The full 8x8 2-D forward transform is given as:

$$Y = TXT^T = \begin{bmatrix} T_{00} & T_{01} \\ T_{10} & T_{11} \end{bmatrix} \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix} \begin{bmatrix} T_{00}^T & T_{10}^T \\ T_{01}^T & T_{11}^T \end{bmatrix} \quad (1)$$
$$= \begin{bmatrix} T_{00}X_{00}T_{00}^T + T_{01}X_{10}T_{00}^T + T_{00}X_{01}T_{01}^T + T_{01}X_{11}T_{01}^T & * \\ * & * \end{bmatrix}$$

As mentioned above, only the coefficients in the upper-left 4x4 block are calculated (the 4x4 sub-blocks that are denoted by "*" in (1) are not calculated and simply set to zero) i.e., we only need to calculate Y_{sub}:

$$Y_{sub} = T_{00}X_{00}T_{00}^T + T_{01}X_{10}T_{00}^T + T_{00}X_{01}T_{01}^T + T_{01}X_{11}T_{01}^T$$

Correspondingly, the inverse pruned transform is given as:

$$X_{sub} = T^T Y T \quad (2)$$
$$= \begin{bmatrix} T_{00}^T & T_{10}^T \\ T_{01}^T & T_{11}^T \end{bmatrix} \begin{bmatrix} Y_{sub} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_{00} & T_{01} \\ T_{10} & T_{11} \end{bmatrix} = \begin{bmatrix} T_{00}^T Y_{sub} T_{00} & T_{00}^T Y_{sub} T_{01} \\ T_{01}^T Y_{sub} T_{00} & T_{01}^T Y_{sub} T_{01} \end{bmatrix}$$

To implement this pruned transform, the existing 8x8 full transform butterfly structure could be re-used with only slight modifications. More specifically, from (1) and (2), we can see that in every 1-D forward (inverse) transform, 4 out of 8 output (input) data is zero and the operation counts in last (first) stage of the transform can be reduced. Take inverse transform as an example. Assume the input and the output of first stage of the transform are d0~d7 and e0~e7 respectively. Using pruned transform, the first stage of the full transform could be simplified because d4=d5=d6=d7=0. Table 1 compares the implementation of the proposed pruned 8x8 transform with the full 8x8 transform in H.264/AVC. Compared to the 8x8 transform in H.264/AVC which needs 32 additions and 10 shift operations for 1-D transform [6], pruned 8x8 transform needs only 20 additions and 7 shift operations.

Table 1. Full 8x8 transform vs. Pruned 8x8 transform

	Full 8x8 transform	Pruned 8x8 transform
e0	d0 + d4	d0
e1	– d3 + d5 – d7 – (d7 >> 1)	– d3
e2	d0 – d4	d0
e3	d1 + d7 – d3 – (d3 >> 1)	d1 – d3 – (d3 >> 1)
e4	(d2 >> 1) – d6	(d2 >> 1)
e5	– d1 + d7 + d5 + (d5 >> 1)	– d1
e6	d2 + (d6 >> 1)	d2
e7	d3 + d5 + d1 + (d1 >> 1)	d3 + d1 + (d1 >> 1)

For transforms which can be implemented using matrix multiplications (those with no right-shifting inside every 1-D transform), the forward and backward pruned transform can be implemented using $\mathbf{T}_{sub}=[\mathbf{T}_{00}\ \mathbf{T}_{01}]$ instead of **T** in (1) and (2) so that complexity can also be reduced.

3.2. Interleaving Process

In the interleaving process, the quantized transform coefficients of 4 4x4 blocks (blocks A_{sub}, B_{sub}, C_{sub}, D_{sub}) are combined into a single 8x8 block (S). This is done as follows. Assume A_{ij} indicate the quantized transform coefficient of block A, where i and j are the horizontal and vertical indices respectively (i.e. A_{00} is the DC coefficient of the block A). The interleaved block S is formed by taking a single coefficient one by one from each block in the order D_{sub}, C_{sub}, B_{sub}, A_{sub}. This kind of reverse scan is used to place the coefficients of the blocks with largest prediction error energy first, so that entropy coding is more efficient for block S. The prediction error of block D is typically the largest because of its longest distance to the nearby predictors. The coefficients are taken from blocks in zigzag order within the 4x4 blocks as shown in Figure 1, and placed in S in 8x8 zigzag scan order. The resulting interleaved block S is given as:

$$S = \begin{bmatrix} D_{00} & C_{00} & C_{01} & B_{01} & B_{20} & A_{20} & A_{03} & D_{12} \\ B_{00} & D_{01} & A_{01} & C_{20} & D_{11} & B_{03} & C_{12} & B_{31} \\ A_{00} & D_{10} & D_{20} & C_{11} & C_{03} & B_{12} & C_{31} & A_{31} \\ C_{10} & A_{10} & B_{11} & D_{03} & A_{12} & D_{31} & D_{22} & C_{23} \\ B_{10} & A_{11} & A_{02} & D_{21} & A_{30} & C_{22} & D_{23} & B_{23} \\ D_{02} & B_{02} & C_{21} & B_{30} & B_{22} & A_{13} & A_{23} & D_{33} \\ C_{02} & B_{21} & C_{30} & A_{22} & B_{13} & D_{32} & A_{32} & C_{33} \\ A_{21} & D_{30} & D_{13} & C_{13} & C_{32} & B_{32} & B_{33} & A_{33} \end{bmatrix}.$$

Coefficients from different components (e.g., two chroma components) can also be interleaved together.

The advantage of interleaving the coefficients into one 8x8 block is that 8x8 entropy coding structure could be reused for coding the block S. This is very important for those standards (or some low complexity encoders) which only support 8x8 transform and entropy coding due to complexity consideration, like the emerging China's Audio and Video Coding Standard [7].

4. COMPLEXITY ANALYSIS

Using the proposed algorithm, encoding and decoding complexity is reduced compared to INTRA_16x16 mode in H.264/AVC mainly because of the following three factors.

1. Transform: Let's consider the luminance component as an example. For a whole macroblock, full 8x8 transform needs 32x16x4=2048 additions and 10x16x4=640 shift operations, while pruned 8x8 transform needs only 20x12x4=960 additions and 7x12x4=336 shift operations, noting that only 12 (rather than 16) 1-D transforms are needed to compute a complete 2-D pruned transform. We also note that full 4x4 transform needs 8x8x16=1024 additions and 2x8x16=256 shift operations. Moreover, second stage DC hadamard transform (which needs 64 additions) is not needed any more.

2. Entropy Coding: Assume Context Adaptive Binary Arithmetic Coding (CABAC), as defined in H.264/AVC is used, and its complexity is reduced as follows. First, the number of CABAC context models is reduced, i.e., context models for original Intra16x16 DC and AC coefficients and Chroma DC coefficients are not needed anymore. In addition, side information needs to be coded for each block, such as EOB (End of Block) and CBF (Coded Block Flag) is reduced. Lower complexity is also achieved in encoding and decoding the significance map and the non-zero transform

coefficients, mainly because fewer coefficients need to be coded.

3. Deblocking: The deblocking filter complexity is reduced due to the less number of edges that need to be filtered with the proposed algorithm. For luminance component, only 8x8 block edges need to be checked if they need to be filtered rather than 4x4 block edges. For chroma components, no edges inside the macroblock need to be filtered anymore.

5. EXPERIMENTAL RESULTS

We have implemented the proposed implementation as described in Section 3 on the H.264/AVC test platform JM 12.4 [8]. Extensive experiments have been performed using the following test conditions: All frames are coded INTRA, using all the available macroblock modes. CABAC is used as the entropy coding method and Quantization Parameter (QP) takes values 25, 30, 35 and 40. Table 2 summarizes the complexity reduction achieved using the proposed algorithm for different test sequences for transform, CABAC and deblocking parts of decoder respectively. The test sequences are chosen, where the INTRA_16x16 mode helps considerably so that we have a fair comparison for the proposed algorithm. It should be noted that the complexity reduction numbers are for the entire sequence including all the modes, and not just measured for the proposed mode.

Last column in Table 2 summarizes the coding efficiency performance in terms of average difference of PSNR (ΔPSNR) of the proposed algorithm according to [9] for the same test set. It can be seen that the coding efficiency penalty using the proposed method is negligible (0.04 dB on average), with significant reduction in complexity. The coding efficiency of the proposed algorithm could be further improved by optimizing the CABAC context models and introducing a 2x2 DC hadamard transform. The numbers in parentheses illustrate the penalty of disabling INTRA_16x16 mode altogether. Compared to disabling INTRA_16x16 mode, the proposed algorithm improves the performance on average by about 0.3 dB.

The complexity reduction for the sequence ShuttleStart is shown in Figure 2 for different tested QPs. It can be seen that the complexity reduction for this case is up to 22% for transform, 12% for CABAC and 39% for deblocking filter.

Table 2 Performance of the Proposed Algorithm

Sequence	Trans.	CABAC	Deblock	ΔPSNR
ShuttleStart	12.87%	7.44%	32.98%	0.01 (-0.60)
Jets	7.20%	4.18%	29.40%	-0.07 (-0.36)
Cyclists	1.69%	2.34%	23.06%	-0.03 (-0.16)
Crew	4.43%	3.52%	30.80%	-0.06 (-0.26)
Average	6.55%	4.37%	29.06%	-0.04 (-0.35)

6. CONCLUSIONS

In this paper, we proposed a novel method optimized for High Definition (HD) video coding utilizing pruned transforms and interleaving multiple transform coefficient blocks. The proposed algorithm provides significant reduction in complexity of intra macroblocks by reducing the number of entropy coded transform blocks and candidate edges for deblocking filter, as well as lowering the complexity of inverse transform by using a pruned transform. Experimental results showed that the complexity of transform could be reduced by up to 22%, entropy coding by up to 12% and deblocking filter by up to 39%, compared to the reference algorithm used in H.264/AVC video coding standard, with practically no penalty on coding efficiency. This makes usage of proposed algorithm very attractive for HD video coding especially for applications and services targeted to power constrained devices, such as mobile phones and portable multimedia players.

7. REFERENCES

[1] J. Ostermann, J. Bormans, P. List, D. Marpe, et al, "Video Coding with H.264 / AVC: Tools, Performance, and Complexity", IEEE Circuits and Systems Magazine, Vol. 4, No. 1, pp. 7-28, April, 2004.

[2] J. Dong and K. N. Ngan, "16x16 integer cosine transform for HD video coding", PCM 2006, Nov, 2006.

[3] J. Dong, K. N. Ngan, C. K. Fong and W. K. Cham, "A universal approach to developing fast algorithm for simplified order-16 ICT", IEEE ISCAS 2007, May, 2007.

[4] S. Naito and A. Koike, "Efficient coding scheme for super high definition video based on extending h.264 high profile," in Proceedings of SPIE VCIP, pp. 607727-1-607727-8, 2006.

[5] N.P. Walmsley, A.N. Skodras and K.M. Curtis, "A fast picture compression technique", IEEE Trans. Consumer Elect., vol. 40, no. 1, pp. 11-19, Feb, 1994.

[6] "Advanced video coding for generic audiovisual services", ITU-T Recommendation H.264, March 2005

[7] L. Yu, F. Yi, J. Dong and C. Zhang, "Overview of AVS-Video: tools, performance and complexity", VCIP 2005, July, 2005.

[8] Available online: http://iphome.hhi.de/suehring/tml/download/

[9] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves", VCEG Doc. VCEG-M33, March 2001

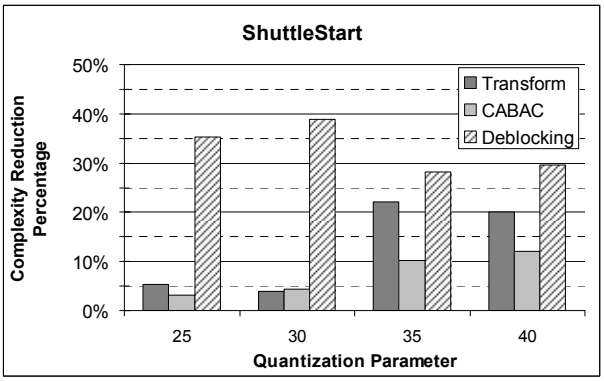

Figure 2. Complexity Analysis for the ShuttleStart Sequence