Transform extension for block-based hybrid video codec with decoupling transform sizes from prediction sizes and coding sizes

Jing Chen[a] and Ge Li[b] and Kui Fan[c]and Xiaoqiang Guo[d]

[a,b,c]School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University
[d]Academy of Broadcasting Science, SAPPRFT, China

ABSTRACT

In the block-based hybrid video coding framework, transform is applied to the residual signal resulting from intra/inter prediction. Thus in the most of video codecs, transform block (TB) size is equal to the prediction block (PB) size. To further improve coding efficiency, recent video coding techniques have supported decoupling transform and prediction sizes. By splitting one prediction block into small transform blocks, the Residual Quad-tree (RQT) structure attempts to search the best transform size. However, in the current RQT, the transform size cannot be larger than the size of prediction block. In this paper, we introduce a transform extension method by decoupling transform sizes from prediction sizes and coding sizes. In addition to getting the transform block within the current PB partition, we combine multiple adjacent PBs to form a larger TB and select best block size accordingly. According to our experiment on top of the newest reference software (ITM17.0) of MPEG Internet Video Coding (IVC) standard, consistent coding performance gains are obtained.

Keywords: Video coding, transform, RQT, IVC

1. INTRODUCTION

Discrete cosine transform (DCT) of different sizes are supported in almost all the current video codecs. However it is usually not mean codecs will spend time to search for finding the best transform sizes for residual blocks. The reason is PBs have various sizes, so codecs need to support several DCT sizes. In general, transform size is same as prediction size for square-shaped PBs, and same as the short side size for rectangle PBs. As a result, this may be not the best choice to make transformation and improve coding efficiency. The behind reasons are that larger transform block sizes, having larger spatial support, provide a better spatial resolution, and smaller transform block sizes, having smaller spatial support, on the other hand, provide a better frequency resolution. Therefore, it is better to select suitable transform sizes for TBs to further improve coding efficiency.

Several approaches have been proposed to solve this problem. In image coding area, the selection of transform sizes is more freely. In the method of Cheng-Tie Chen, 4 adjacent NxN blocks will be merged into one 2Nx2N block for transform based on a mean-based decision rule.[1] Its'Hak Dinstein et al. cluster the basic blocks of the whole images into different sets according to their ac coefficients, then combine blocks in the same set to large blocks which are named superblock.[2] For the video codec standard such as HEVC, H.264, AVS2, AV1, the common method is to divide big TBs into small ones. A concept named adaptive block-size transforms(ABT) is proposed for H.264/AVC. The basic idea is to align the transform size to the prediction size for inter blocks and employ variable sizes for intra blocks.[3] For the newest international video coding standard HEVC, the RQT has been adapted as continuation of prediction quad-tree .[4] The RQT has brought about 1.2% coding efficiency improvement on the HEVC reference software. Via allowing transform spanning two PBs, experimental results showed that about 0.4-0.7% coding efficiency gain will be obtained in HEVC.[5] Chen et al. propose a pre-filtering technique including the combination prediction and the motion vector refinement, to smooth the PB boundaries in order to promote the efficiency of transform spanning PBs.[6] In the Googles open-source project WebM Project, a new transform tool named super-transform have been adopted in VP10.[7] What super-transform does is to create a new predictor based on recursive application of the pre-filtering technique mentioned above.[8]

Among all these methods, those proposed by Cheng-Tie Chen and Its'Hak Dinsteinare are hard to be applied in video coding, since blocks in a video frame are always coded in a raster order. It's not realistic to combine blocks that are far

Further author information: Send correspondence to E-mail: chenjing15@pku.edu.cn, geli@ece.pku.edu.cn, kuifan@pku.edu.cn, guoxiaoqiang@abs.ac.cn

away from each other for transform. ABT and RQT only consider transform sizes smaller than prediction sizes. Although Chen et al. take the transform size bigger than prediction sizes into account, they only use it in some limited prediction situation.

In this paper, we extend the transform via allowing multiple PBs in one TB. At the same time, considering some blocking artifacts may exist along the boundaries of PUs in the residual signal, we also adopt a boundary filter method based on Overlapping Block Motion Compensation(OBMC) to improve the efficiency of big size transform.[9] OBMC is an inter tool which can get a more precise prediction result by making use of neighboring motion information. In general, OBMC is applied in motion compensation, it will compute a weighted prediction value for boundary pixels.[10] As integer transforms of 16x16, 8x8 and 4x4 are supported and adaptively switched on macroblock or block level. ITM 17.0 is a good platform for our experiment.[11] Experimental results show a better coding efficiency can be achieved by involving the larger transform spanning multiple PUs. The rest of the paper is organized as follows. A brief outline of RQT structure and transform spanning two rectangle PBs is reviewed in section 2. The extended RQT is illustrated in section 3. The experimental results are given and discussed in Section 4. Finally, the paper is concluded in Section 5.

2. REVIEW OF THE RQT STRUCTURE AND TRANSFORM SPAN TWO PBS

The partition of a given PB into TBs is done based on a quad-tree approach, its corresponding structure is called the RQT.

2.1 RQT

As the Figure 1 shows, a CB is partitioned into 16 TBs. The individual blocks are processed in numerical order, corresponding to a depth-first tree traversal as shown in the right of Figure 1. In the Figure 1, the RQT structure is shown using the red broken lines.

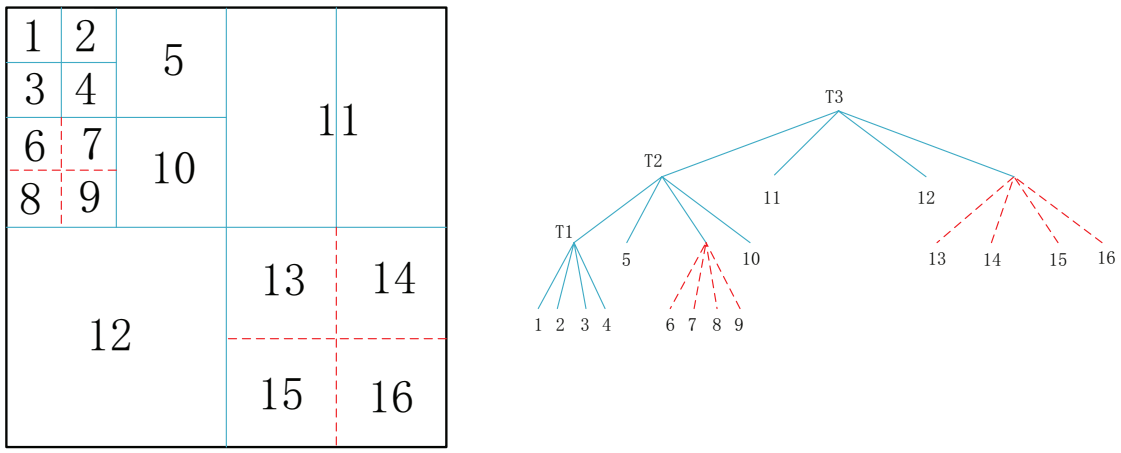

Figure 1. A typical partition of a big coding block, the blue solid lines are the PB boundaries, while the red break lines are the TB boundaries. The number indicates the coding order of the TBs(left), and its partition quad-tree(right)

To find best block sizes for prediction, the CB is divided by the blue solid lines. To find best block sizes for transform, the partitioned blocks are divided further more by the red broken lines. It should be noted that only square block transform units are supported. The quad-tree approach enables to adapt the transform to the varying space-frequency characteristics of the residual signal. The maximum allowed depth of the RQT restricts the number of subdivision levels.

2.2 Spanning transform under restrictions

The RQT split big transform blocks to small ones. It shows that there are also rich high frequency components in some residual blocks. However, some residual blocks are spatially stationary so that a larger transform size will have a better compression efficiency. Then our question is that what kind of PBs can be selected for the check of big size transform. Firstly, they should be inter PBs. Because for intra PBs, the causal neighboring blocks (i.e. those blocks which are encoded and transmitted before the current block in processing order), have to be fully reconstructed in order to be able to generate

the prediction signal for the current block. So for the intra case, there is a forced subdivision of the PB into sub TBs, such that none of these sub TBs cover more than one PB. For inter PBs, every block is predicted independently. So there is no problem when TBs cover multiple PBs. In some special cases, TBs can span intra PBs when the intra PBs do not depend on other PBs within the same TB, that means the only intra PBs is located in the top or the left corner. Secondly, they should adjacent to each other. We should ensure this technology is compatible with the current video coding framework.

The large transforms can provide a better energy compaction and a better preservation of detail in a quantized signal than a small transform does. On the other hand, larger transforms may introduce more ringing artifacts caused by quantization than small transforms. As we known, a PB has only one motion vector for all its pixels. Generally speaking, PBs with different motion vectors have great difference in their image contents. However, the situation is not the same when it comes to their boundaries. Imagine that there are two adjacent PBs, the pixels in their boundaries have the same values. But with the impact of two different motion vectors, great difference will come into being in the boundary's two sides. Than if we do bigger transform spanning the two PBs, some unnecessary high frequency coefficients will be produced. Connected with the quantized error, ringing artifacts will be produced. That is the worst situation.

So when it comes to our research of the transform spanning PBs, great attention has been attained to mitigate the discontinuity produced by over-simplified motion compensation. Chen et al. are inspired by the method of the OBMC. They propose a pre-filtering technique that combine the two motion vectors of the two PBs in a weighted way to get predict values. The weight value is decided according to the distance from the pixel to the boundary. And the weight function is not linear. The other technique they proposed is motion vector refinement. The motion vectors are generated separately on standard encoding process first, then a final step to refine the motion vector for the optimal filtered prediction.

3. THE EXTENDED RQT AND THE WAY TO IMPROVE THE EFFICIENCY

3.1 PB combination for big TB

Our extended RQT can be described as following steps. Step 1:divide the macroblock for prediction purpose; Step 2: get the residual blocks and then treat them as a big residual macroblock; Step 3: divide the residual macroblock for transform and quantization; Step 4: get the coefficients to encode. By doing block division twice, it realizes the decoupling of the transform sizes and prediction sizes.

When it comes to engineering implementation, we may observe that the second division–transform division have been done one half by allowing the division of PBs, as the traditional RQT does. What we need to do is to try transform that span multiple PBs. As we restrict the TBs to square blocks, we may find that we only need to check out the partitions that are labeled by T1–T3 in the right quad-tree of Figure 1 for example. T1–T3 are non-leaf nodes, indicating that the transform will span multiple PBs. If we define the root node as level 0, then the level difference between a node and its leaf children will decide how many PBs the transform will span. The difference between traditional RQT and the proposed extended RQT is their different starting point. The tradition way starts at the smallest square PBs, such as the block 1,5,12 in Figure 1 On the other hand, our method starts at the basic coding block, which include the block from 1 to 16 in Figure 1.

In codecs, we encode a block to bit stream in the sizes of PBs. However, if we do transform that span multiple PBs, we have to encode the block in the sizes of transform. Then we will encode a block with more than one motion vectors, just as we encode a block with coefficients that resulting from different transform sizes in the traditional RQT.

By checking few big size transform, the extended RQT realizes the goal to find best transform sizes only according to the space-frequency character. What's more, its change from a modern mainstream codec framework is very miner. We only need to signal the decoder a new type of coding blocks with different motion vectors in its different areas. In some codecs, this is even not a new type of coding block.

We also extend the transform to the size bigger than the size of macroblock (or coding tree unit in HEVC). If the size of macroblock is 16x16, we may check the transform in 32x32 size for four macroblocks. In essence, this method achieves the same effect as extending the macroblock size to 32x32 if we don't do prediction in the whole macroblock size. From the point of engineering realization, extending the transform size may be easier than extending the macroblock size in some codec platforms. Firstly, we need to change the encode order of macroblock. In the mainstream codecs, macroblocks are encoded in raster scan order. If we want to do transform in the size twice of the macroblock size, we realize that it is essential to partition the frame in size of 32x32. As shown in Figure 3, a small rectangle is a macroblock. In the right

part of the Figure 3, we may suppose macroblock 1, 2, 3, 4 belong to a big macroblock. The encoding order is raster scan for big macroblock, following raster scan for macroblocks in this big macroblock. For some macroblocks that are in the boundary of a frame, there are not enough neighbor blocks to combine a big TB, we will keep its encode order unchanged, as the macroblocks in the third row of the Figure 3 show. After we have encoded the last macroblock in a big macroblock, we check the transform in size two times of macroblock size. If the big size transform has a better coding efficiency, when we encode the block into bit stream, we still adopt the size of macroblock, and encode all the transform coefficients in the first macroblock. The transform coefficients of the last three macroblocks are all set to 0. This method can be used both for luminance and chroma components. For every four macroblocks, one bit indicating whether transform spanning

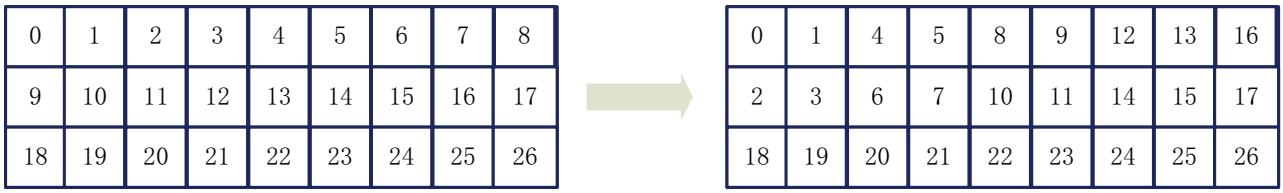

Figure 2. change of the encode order in extended transform, the number labels the encode order

macroblock is selected will be written. The availability of neighbor blocks for prediction reference will also change for some macroblocks. For every macroblock, by allowing transform spanning multiple PBs, more transform sizes for residual are available. A correct RDO process will be essential. For those image areas that become flat after prediction, big size transform will be a good choice with no doubt. In the next section, we will further expound how to improve big transform efficiency.

3.2 Improve big transform efficiency

In the selection process of transform sizes, we tend to select big size transform for the reason we have mentioned in 2.2. So in our proposed method, how to improve the efficiency of big size transform is also very important. The method that reduces high-frequency component is no doubt effective. Chen etc. have proposed a pre-filtering technique to smooth the boundary. And we also think an improved smooth method is needed since we make the transform across PB boundary more complex. The prediction value of all pixels in a prediction block is determined by one Motion Vector(MV). This prediction method may cause the discontinuity in PB boundaries, thus bringing more high frequency coefficients in transform. If we want to avoid this situation, further dispose for boundary pixel prediction is necessary. As we have analyzed, a TB in the extended RQT may have several motion vectors that resulted from separate predictions. For a PB, if the MV of its neighbor block is available, combine its own MV with that MV to smooth their boundary pixels. As Figure4 shows, firstly we split PB boundary pixels into 4x4 blocks. Then processing every 4x4 boundary block as follows: (1) checking if the MV of its neighbor blocks is available and are not identical to its own motion vector; (2) using all motion vectors valid to derive prediction block; (3) using specified weighting factors to obtain the final prediction signal. As the Figure4 shows, for a block with NxN prediction form, the small blocks labeled by shadow are their boundary 4x4 blocks that need to be processed before extended transforms. For block 1—the 4x4 block in the top left corner of the fourth PBs, its above and left neighbor 4x4 blocks are valid as they have different MVs. The below and right 4x4 blocks are not valid as they are in the same PB.

For blocks 1, prediction values can be calculated by

$$p(x,y) = \sum_{i=1}^{4} w_i(x,y) p_i(x,y). \tag{1}$$

Where i denotes the number of neighbor blocks, $p_i(x,y)$ denotes the value of prediction block based on motion vector of block i, and w_i is its weight factor. If the neighbor block is not valid, the weight factor will always be 0;

As for how to decide the value of non-zero weight factor, we may assign some constant since the block size to do OBMC is just 4x4. In our experiment, the weighting factors 1/4, 1/8, 1/16, 1/32 are used for prediction block based on motion vectors of a neighboring block and the weighting factors 3/4, 7/8, 15/16, 31/32 are used for the prediction block based on motion vectors of the current block. The weighting factor of one pixel is decided by its distance from the boundary line.

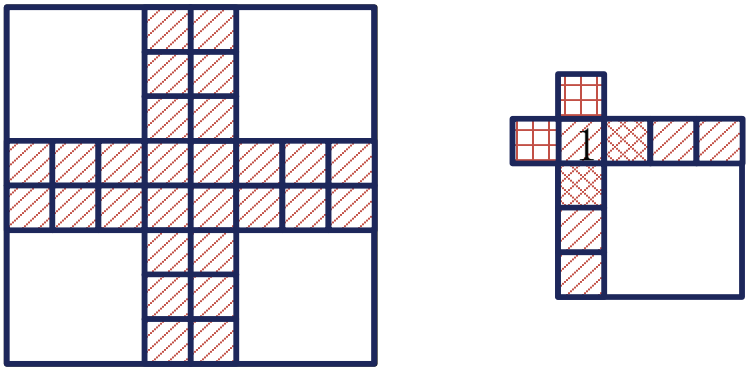

Figure 3. blocks need to do OBMC(left) and 4x4 blocks and its neighbor blocks(right)

For pixels in 4x4 blocks, the range of their distance from the boundary line is from 1 to 4, corresponding to the weighting factors mentioned above. We use this method to process the PB boundaries, thus achieving the goal to smooth the big transform block, reducing high frequency coefficients and improving the coding efficiency.

4. EXPERIMENTAL RESULTS

To verify the performance of the proposed algorithm, we have conducted experiments on the newest reference software (ITM17.0) of MPEG Internet Video Coding (IVC) standard. The experiment follows the IVC standard test condition .[12] The sequences used in the experiment are shown in table 1, and QP is set to be 27, 32, 38, and 45. In the experiment, sequences are tested under Random-Access (RA) configuration. The experimental results are shown in Table 1.

Table 1. Performance of the proposed method on ITM 17.0

Class	**Sequence**	**BD_rate[%]**		
		Y	U	V
UHD 2560x1600	PeopleOnStreet	-0.8	-1.5	-2.0
1080p (1920x1080)	ParkScene	-0.6	-0.2	-0.9
WVGA (852x480)	PartyScene	-1.8	-2.2	-2.0
	BasketballDrill	-2.4	-1.7	-1.6
	BQMall	-0.7	-0.8	-0.8
	RaceHorsesC	-2.2	-1.5	-1.7
WQVGA (416x240)	BasketballPass	-1.8	-2.1	-0.9
	BQSquare	-0.3	-0.1	-0.3
	BlowingBubbles	-1.3	-4.2	-4.3
	RaceHorses	-1.8	-1.2	-1.3
720p (1280x720)	FourPeople	-0.7	-0.6	-0.5
	Johnny	-1.7	-1.5	-0.6
	KristenAndSara	-1.0	-1.2	-1.0
Total Average		**-1.1**	**-1.2**	**-1.0**

We also test the proposed transform extension on some high resolution sequences. The result is shown in Table 2.

Table 1 shows that our proposed method have an average 1.1% gain for luminance component. Table 2 shows that for some high resolution sequences the frames of which are relative rest, the extended transform can obtain coding efficiency gain without OBMC.

Table 2. Perfornance of the proposed method on ITM 17.0 for high resolution sequence

Class	Sequence	Extended Tansform			Extended Tansform & OBMC		
		BD_rate[%]			BD_rate[%]		
		Y	U	V	Y	U	V
1080p (1920x1080)	ParkScene	-0.2	0.3	-0.2	-0.6	-0.2	-0.2
	taishan	-0.7	-1.8	-1.5	-2.0	-1.6	-1.7
720p (1280x720)	FourPeople	-0.5	0.1	0.2	-0.7	-0.6	-0.5
	Johnny	-1.1	-0.3	0.3	-1.7	-1.5	-0.6
	KristenAndSara	-0.9	-0.6	-0.4	-1.0	-1.2	-1.0
Total Average		**-0.7**	**-0.4**	**-0.3**	**-1.2**	**-1.0**	**-0.8**

5. CONCLUSION

In this paper, an extended transform method is proposed. In addition to the traditional transform sizes that are equal to or smaller than the corresponding PUs, big TBs by combining multiple PBs are considered in the proposed method. For alleviating the blocking effects at the PUs boundaries, we perform OBMC before the extended transform is applied. Experiments show that, average 1.1% coding performance gains are obtained on ITM17.0. Future work includes improving coding efficiency of big size transform based on prediction optimization.

ACKNOWLEDGMENTS

This work was supported by the grant of National Science Foundation of China (No.U1611461), Shenzhen Peacock Plan (20130408-183003656), and Science and Technology Planning Project of Guangdong Province, China (No. 2014B090910001 and No. 2014B010117007).

REFERENCES

[1] C.-T. Chen, "Adaptive transform coding via quadtree-based variable blocksize dct," in *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*. IEEE, 1989, pp. 1854–1857.

[2] I. Dinstein, K. Rose, and A. Heiman, "Variable block-size transform image coder," *IEEE transactions on communications*, vol. 38, no. 11, pp. 2073–2078, 1990.

[3] M. Wien, "Variable block-size transforms for h. 264/avc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 604–613, 2003.

[4] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, "Block partitioning structure in the hevc standard," *IEEE transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1697–1706, 2012.

[5] W.-J. H. Tammy Lee, Jianle Chen, "Joint collaborative team on video coding (jct-vc), document jctvc-c200,te 12.1: Transform unit quadtree/2-level test," 2010.

[6] Y. Chen, K. Rose, J. Han, and D. Mukherjee, "A pre-filtering approach to exploit decoupled prediction and transform block structures in video coding," in *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4137–4140.

[7] S. Parker, Y. Chen, J. Han, Z. Liu, D. Mukherjee, H. Su, Y. Wang, J. Bankoski, and S. Li, "On transform coding tools under development for vp10," in *SPIE Optical Engineering+ Applications*. International Society for Optics and Photonics, 2016, pp. 997 119–997 119.

[8] D. Mukherjee, H. Su, J. Bankoski, A. Converse, J. Han, Z. Liu, and Y. Xu, "An overview of new video coding tools under consideration for vp10 the successor to vp9," in *SPIE Optical Engineering+ Applications*. International Society for Optics and Photonics, 2015, pp. 95 991E–95 991E.

[9] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 693–699, 1994.

[10] R. W. T. H. W. G. Hao Li, Kui Fan, "Iso/iec jtc1/sc29/wg11 mpeg2017/m39997: Obmc for ivc+," 2017.

[11] R. Wang, G. Li, S. Park, J. Kim, T. Huang, E. Jang, and W. Gao, "Overview of mpeg internet video coding," in *SPIE Optical Engineering+ Applications*. International Society for Optics and Photonics, 2015, pp. 95 991H–95 991H.

[12] R. Wang, Z. Wang, K. Fan, T. Huang, W. Wang, G. Li, and W. Gao, "Mpeg internet video coding standard and its performance evaluation," *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.