AFRL-AFOSR-VA-TR-2019-0101

Interframe JPEG-2000 for Motion Imagery Browsing

John Woods
RENSSELAER POLYTECHNIC INST TROY NY

12/14/2018
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTA2
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
14-04-2019	Final Performance	01 Sep 2014 to 31 Aug 2018

4. TITLE AND SUBTITLE
Interframe JPEG-2000 for Motion Imagery Browsing

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-14-1-0236

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
John Woods

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
RENSSELAER POLYTECHNIC INST TROY NY
100 8TH STREET
TROY, NY 12180 US

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF Office of Scientific Research
875 N. Randolph St. Room 3112
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR RTA2

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-VA-TR-2019-0101

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this research we start with the enhanced MC-EZBC highly scalable
video coder and revise and extend its motion-compensation temporal
filtering (MCTF)framework to incorporate advanced mechanisms such as
motion-vector predictor competition, block-merging, affine
motion-compensation modes, and two-component blocks. We also provide
a new coder backend based on JPEG 2000 to facilitate multilevel
code-stream extraction and gain compatibility advantages with JPEG 2000
image coding. Experimental results showed their ability to both
significantly improve the PSNR performance and also extend the scenario
f scalable video coding. The new coders are called Interframe EZBC and Interframe JPEG 2000.Experimental results showed their ability to both
significantly improve the PSNR performance and also extend the scenario
of scalable video coding. The new coders are called Interframe EZBC and Interframe JPEG 2000.

15. SUBJECT TERMS
JPEG, Image browsing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF

a. REPORT	b. ABSTRACT	c. THIS PAGE		PAGES	19a. NAME OF RESPONSIBLE PERSON
Unclassified	Unclassified	Unclassified	UU		NGUYEN, TRISTAN
					19b. TELEPHONE NUMBER *(Include area code)* 703-696-7796

An Improved Video Coding Framework with High-scalability: A Final Report on AFOSR Grant FA9550-14-1-0236

John W. Woods and Yuan Liu

ECSE Department, Rensselaer Polytechnic Institute,
JohnWoods@ieee.org,liuy33@rpi.edu,

Abstract. In this research we start with the enhanced MC-EZBC highly scalable video coder and revise and extend its motion-compensation temporal filtering (MCTF) framework to incorporate advanced mechanisms such as motion-vector predictor competition, block-merging, affine motion-compensation modes, and two-component blocks. We also provide a new coder backend based on JPEG 2000 to facilitate multi-level code-stream extraction and gain compatibility advantages with JPEG 2000 image coding. Experimental results showed their ability to both significantly improve the PSNR performance and also extend the scenario of scalable video coding. The new coders are called Interframe EZBC and Interframe JPEG 2000.

1 Introduction

The development of video coders during the last ten years has seen considerable improvement in coding efficiency due to more powerful motion-compensation (MC) strategies that efficiently exploit correlation between frames as well as more powerful frame-residual coding methods. While standards-based coders such as AVC/H.264 [1] and HEVC/H.265 [2] were designed to achieve a single target bitrate, frame rate, and resolution, modern scalable coders have attracted attention from researchers for their capability of extracting and decompressing multiple versions of coded video according to the various requirements of subscribers. Moreover, for a scalable coder, more accurate but time-consuming compensation and compression methods can fit into their one-time encoding process to achieve improved performance on multiple decodings. This high scalability can facilitate video browsing which is important in the surveillance application areas.

Based on the fine-grain scalable embedded zero-block coder (EZBC)[19], Wu et al. [20] presented the motion-compensated video codec enhanced MC-EZBC. A *"t+2D"* decomposition process was employed to perform multi-level temporal decomposition. While enhanced MC-EZBC has proven to be an effective, highly-scalable video coder, still some modern developments over the last 10 years attracted our attention and inspired us to revise this already efficient video coder:

First, the MCTF framework of enhanced MC-EZBC has become outdated compared with the state-of-the-art development of video compression techniques. The translational motion compensation mechanism, median motion vector (MV) prediction system and corresponding $DIRECT$ mode cannot sufficiently detect redundancy among consecutive video frames are not state-of-the-art. New mechanisms such as higher-order motion modes, motion estimation based on competitive MV predictor selection, and motion-block merging have recently shown significant improvement in coding efficiency, and can be incorporated into the EZBC video coder.

Second, the performance of other subband/wavelet coders in encoding video temporal-high and temporal-low frames after MCTF processing also attracts wide interest of researchers. Regarding our fully scalable coding structure, the EZBC subband coder has some limiting issues in its present software realization related to multiple adaptations or scaling in a hierarchical cache-based distribution scheme [3] that may require several successive code-stream extractions from the master bitstream. Our current enhanced MC-EZBC coder software realization only allows a single extraction from the original encoded bit file.

Third, block-based motion estimation allows only one motion pattern for pixels within each block, no matter it is translational or higher-order ones. However, a complicated motion boundary may require considerable quadtree decomposition steps resulting in many small motion blocks and redundant motion bytes to encode in a traditional estimation system. A powerful new idea is to allow a motion boundary inside a single block rather than the conventional alternative (think H.264/AVC)of introducing many small blocks to describe the motion edge or boundary. Certain auxiliary methods may be needed to determine the best matching for two moving areas within such a block. Some papers [35][36] have shed light on the possibility of detecting and encoding two or more motion patterns within one motion block, which may significantly improve coding efficiency and reduce the complexity of the resulting quadtree structure.

Fourth, a scalable coder estimates and encodes the Interframe MV information with a certain Lagrange parameter λ_m. However, this parameter is highly affected by the nature of the motion activity as well as the bits available for subband coding after MCTF based on the targeted range of scalable bitrates desired. So adaptively modifying λ_m could potentially allow a fully scalable coder to further enhance its coding performance.

Based on these observations, this research concentrated on revising the enhanced MC-EZBC video coder in both motion estimation/compensation and subband-coding aspects. Additionally the underlying subband/wavelet frame coding method was geneeralized to include the possibility of using JPEG 2000 in place of EZBC. The improvement of coding performance and extension of scalability of the two new interframe scalable video coders will be compared and discussed with that of the existing enhanced MC-EZBC coder and the popular standards-based nonscalable video coders H.264/AVC and H.265/HEVC.

2 Background

This section will provide necessary background on the highly scalable MC-EZBC video coder and then introduce improvements in motion vector coding based on competition, review the relatively new concept of motion-block merging for enhanced motion data compression, and then introduce a generalization of the motion representations from translation to affine motion, that can better capture rotations and zooms. Then we bring in the JPEG 2000 method for video coding, followed by enlarging motion possibilities to include more than one motion vector per block. Together with merging, two-component motion per block can provide a very efficient representation of translational motion in the video. Finally we introduce the need to determine the balance of motion data and video prediction residual information in a real video coder. In our framework this is done via choice of Lagrange multiplier $lambda_m$.

2.1 Enhanced MC-EZBC Coder

A typical example of *"t+2D"* temporal decomposition in the enhanced MC-EZBC coder is illustrated in Fig. 1. The first step is MCTF processing of a group of pictures (GOP), typically consisting of 16 consecutive video frames. As illustrated, the information within about half of these frames is predicted or compensated from their neighbor frames on each side (forward and backward), thus generating high-frequency temporal subbands (vertical dashed-line arrows) that contain only limited prediction-error or residual information. Temporal-low frames (vertical solid arrows) are also created from the assumed motion field. It is important to note that these temporal low frames make up the lower frame-rate scaled versions for this coder. The spatial indices to retrieve redundant information from neighbor reference frames are called motion vectors (MV), which reflect the displacement of moving objects among different frames. Continuing this decomposition for a GOP of length 16, a multi-level temporal decomposition is be created with MV sets 1-34 (as denoted by horizontal arrows) as shown, where the existence of MV sets 1,3,4,8,9,17,18 and 34 depend on the GOP boundaries are *open* or *closed* for bidirectional compensation. At the receiver, the inverse decoding process can restore the original information by receiving only the temporal-low subband at the highest temporal level (lowest frame rate) together with all temporal-high subbands, and the relevant MV sets.

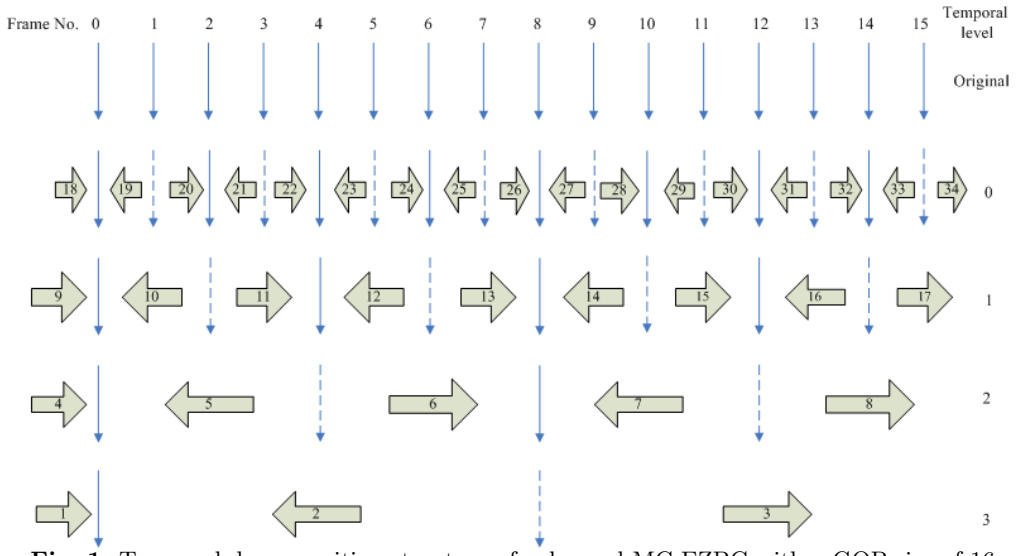

Fig. 1. Temporal decomposition structure of enhanced MC-EZBC with a GOP size of 16

To create these MV sets, enhanced MC-EZBC uses a quadtree block-based motion estimation system, with the largest block called the macroblock. Each macroblock then goes through a series of quadtree structure splits based on an operational rate-distortion (R-D) competition between each block and its four child blocks, wherein every macroblock becomes the root node of a quadtree. Similar strategies have been widely applied in standards-based coders such as H.264/AVC and H.265/HEVC [22] as well as in scalable coders [20]. For each block, an effective MV search is done on both itself and four equal-sized child blocks split from it. Based on the estimated bit cost to encode motion-vector information of each block

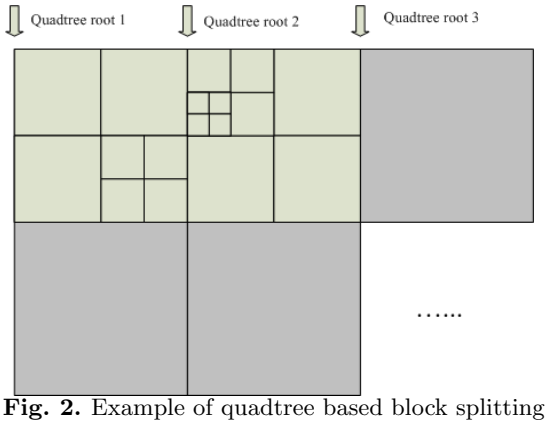

Fig. 2. Example of quadtree based block splitting

and the distortion value of compensation measured in the sum of absolute difference (SAD), [9] gave the R-D cost based on motion-estimation residual as:

$$J_{mv} = \lambda_m \cdot R_{mv} + D_{DFD}, \tag{1}$$

where D_{DFD} denotes the displaced frame difference (DFD) error, i.e., the distortion obtained from interframe and intraframe motion compensation. The R_{mv} includes all bits needed for coding the prediction information. A Lagrange multiplier λ_m is used as the key coefficient to balance the weights of distortion and motion bits, which can be specified in the configuration file of the enhanced MC-EZBC coder. Comparison between the R-D cost of such block and the corresponding R-D cost sum for its child blocks will decide which ones should be kept to optimize overall performance. Such splitting processes result in the irregular quadtree structure as shown in Fig. 2. In the implementation, the motion estimation of quadtree blocks is performed according to a so-called z-scan order, which allows each one to utilize spatial neighbor motion information that has been created.

With a certain overall or total bitrate as the target, a further overall R-D cost equation [9] can be given as:

$$J = \lambda \cdot R_{tot} + D, \tag{2}$$

where R_{tot} denotes the total bits for coding video frames, including those for transmitting motion compensation information as well as coding estimation residual and temporal-low subbands, and D refers to the distortion of reconstructed video as compared to the original one. Experimentally it is known that a positive correlation exists between the overall Lagrange multiplier λ and the motion-estimation one λ_m.

Of course, a scalable coder has no precise target for total bitrate, still with optimizing a scalable coder for best performance over a range of bitrates in mind, we can see that a corresponding range of λ_m would be needed. As a practical matter, a fixed value of λ_m is used to provide best overall performance over a desired range of total bitrates. Practically speaking, this suffices when the desired target range for total bitrate is not too large. Otherwise, some scalable coding for motion would also be required.

The enhanced MC-EZBC coder employs both interframe and intraframe compensation methods for its quadtree blocks. Table 1 shows the features of all enhanced MC-EZBC compensation modes, including the mode number in the coding system, the reference frame direction for MCTF, with or without the update step for temporal-low subbands and MV

Table 1. Motion compensation modes for enhanced MC-EZBC [15]

Mode No.	Block mode	MCTF direction	Update step	MV residual
1	bi-connected	BI	Yes	2
2	left-connected	LEFT	Yes	1
3	right-connected	RIGHT	Yes	1
4	bi-predicted	BI	No	2
5	left-predicted	LEFT	No	1
6	right-predicted	RIGHT	No	1
7	parallel	BI	Yes	1
8	spatial-direct	BI	Yes	0
9	I-block	INTRA	No	0

residual counts, respectively. Among them, modes 1-6 are ordinary ones with or without update step in MCTF lifting [18] (which is determined by the motion prediction error and a given threshold). PARALLEL mode means the backward MV can be directly inferred from the forward counterpart to save motion information bits, SPATIAL-DIRECT mode will directly obtain motion vectors from neighbor blocks to save MV residual bits, and I-BLOCK is the intra mode that resembles the counterpart used by H.264/AVC. This last mode is used when interframe motion estimation has failed, i.e. has not produced a useful "compensation." The I-BLOCK then resorts to inframe or spatial prediction for its attempt to compensate the current block.

2.2 Competition-based Motion Vector Predictor Selection

To improve the efficiency of interframe compensation as well as motion-data coding, the motion estimation of each block always starts with an MV predictor, and then follows up with a DPCM-like coding strategy by searching and encoding the MV residual to further enhance the rate-distortion performance [6] for the current block. As a result, a motion vector residual needs to be encoded:

$$e_{mv} = mv - mv_p, \tag{3}$$

where mv_p is the motion vector predictor (MVP) and mv refers to the measured motion. For a typical interframe compensation based on a gradually changing optical flow [7], the bits needed for encoding motion data can be significantly reduced in this way.

One of the widely applied MVP obtaining strategies is based on median prediction that uses the MVs of previously processed spatial neighbor blocks to create an MVP as the median value of these neighbor MVs. This method works fast and requires no additional bits for encoding any index for the median predictor, and was chosen for both H.264/AVC [1] as well as enhanced MC-EZBC [20] for video coding.

However, Laroche et al. [11] revealed the potential performance improvement obtained by selecting an optimal MV predictor from available neighbor candidates to further optimize rate-distortion performance and then proposed a *competition-based* MVP selecting method. Here an index needs to be transmitted to the decoder to let it know which predictor is selected. Possible predictor candidates include both prior processed spatial neighbor blocks in the current frame as well as collocated blocks in prior processed neighbor frames, as shown in Fig. 3 borrowed from [11].

Here the $mv_a - mv_d/mv_{col}$ are potential spatial/temporal candidates to obtain the MVP for the current block. To reduce computational complexity for a non-scalable coder, Laroche

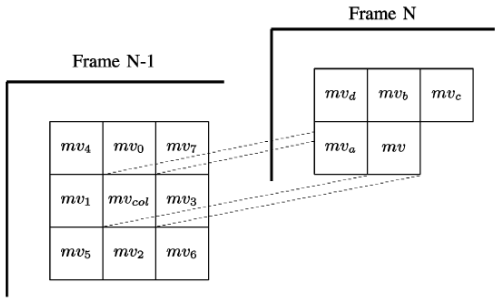

Fig. 3. Example of MV Predictor Candidates in [11]

et al. only perform a similar MV search based on median predictor like H.264/AVC, and then compare the cost of residual and MVP index coding bits based on each candidate in the minimization, but no new motion estimation is performed. For H.264/AVC there is also a $SKIP$ mode where the MVP candidate which results in the best Lagrange performance is selected and no coding of the prediction residual is employed.

Such competition-based MVP selection must be identical on both encoder/decoder sides so that the same prediction information is obtained to correctly construct the MV for the current block, thus the neighbor information should come from the neighbor frames/blocks that have already been encoded/decoded before current block. Such a competition-based MVP selection strategy finally resulted in the advanced motion vector prediction (AMVP) scheme of H.265/HEVC [8], which sets up a candidate list with a maximum length of 5 and considers candidates from 5 spatial neighbor positions, 2 collocated positions of the previous frame, with their median combination as a supplement. Reference [11] shows that 2% to more than 10% of bitrate savings can be achieved by applying such a competition-based MVP mechanism on a range of test clips.

2.3 Motion Block Merging

The spatial direct mode was widely applied in earlier coders such as H.264/AVC and enhanced MC-EZBC, which allows translationally interframe compensated blocks to directly obtain motion information from their spatial/temporal neighbors without searching for and transmitting any additional MV residual, thus significantly reducing the time cost for motion search as well as bitrate cost for transmitting residual.

However, later researchers [25] found that these neighbor blocks may belong to the same object within a video scene and thus share exactly identical motion patterns, which is referred to as mutual "merging" of motion blocks. Although a spatial direct mode may has similar effect, the widely existing multi-reference frame compensation modes tend to select direct MV information from different sources. Compared with direct modes, merging could further reduce the MVP index coding bits by letting blocks mutually inherit motion info from neighbor merging target block and thus form a "merged region" that has constant motion patterns, including the motion mode (forward, backward or bidirectional) and MV values. Such new thinking finally resulted in the development of various $MERGE$ modes.

Matthew et al. [26] implemented a "leaf merging" method, which detects the motion similarity between compensated blocks and forms multiple merging areas that share the same MV information. For each merged region, only the motion data of one block will be transmitted, such block is referred to as the "anchor node" which records the motion pattern

of the region. However, such a merging mode is implemented in an additional raster-scan detection round right after the traditional motion estimation and may violate the mutual dependence of MV prediction information based on z-scan order, thus requires further post-processing to modify them when necessary.

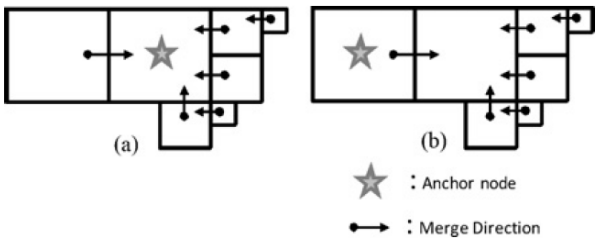

Fig. 4. Example of Leaf-Merging in [25]

Later, Helle et al. [22] proposed a block-merging mechanism, where a similar neighbor block motion information inheritance like [26] is generated, but is performed in a causal way so that no additional round of processing is needed. For each current MC block, the spatial and temporal neighbor blocks' motion information created before it can be taken as the possible merging target, thus the best merging pattern can be obtained and compared with other motion-compensation modes based on R-D optimization to make a decision. This research finally resulted in the inter-picture prediction block merging mode in H.265/HEVC, as shown in Fig. 5 taken from [8].

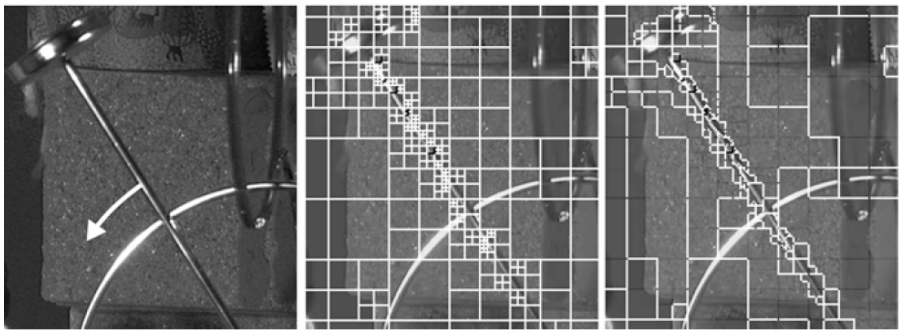

Fig. 5. Outcome of H.265/HEVC Block Merging in [8]

Due to the similarity between spatial direct and block-merging mode in processing the translational motion model, in the working draft 2 ($WD2$) of H.265/HEVC, the $DIRECT$ mode as still employed in old working draft $WD1$ [10] was finally replaced by a $BLOCK\ MERGING$ mode. However, the essential idea of "merging" a current block into neighbor blocks motion field should not just stop at translational motion, higher-order motion models or other special compensation mechanism may also employ the idea of "merging" to enhance their performances.

2.4 Affine Motion Model

Traditional video coders perform compensation with only translational motion, which can be relatively easy to implement. However, more complicated motion patterns such as zoom or rotation cannot be effectively simulated by translational model without requiring many small blocks within those regions. Some researchers then moved on to higher-order motion models to effectively compensate those patterns. Among these, the affine motion model [27] attracted wide attention.

Early researchers [27][28] introduced affine mesh-based models to represent this more complex motion, where the compensated frame is divided into multiple triangular mesh cells, then each cell will be further divided into two triangular patches as shown in Fig. 6 borrowed from [29], including two options known as UL and LU, whereby each triangular patch has three corner-point control MVs as shown in the figure to define the affine motion within it. For the lower triangle of LU partition of a $s \times s$ block, the pixel-wise MV (v_x, v_y) can be denoted as:

$$\begin{aligned} sv_x &= (s-y)v_{x1} + (y-x)v_{x3} + xv_{x4}, \\ sv_y &= (s-y)v_{y1} + (y-x)v_{y3} + xv_{y4}. \end{aligned} \tag{4}$$

Here (x, y) denotes the coordinate of arbitrary pixel with respect to the upper-left corner of the block. However, each control will affect all neighboring triangular mesh grids by imposing a continuous affine motion pattern, which is not fit for suddenly changing motion boundaries and may waste bits in coding those translational motion regions.

A causal three-control-point affine model based on existing MC blocks was then introduced in [24] with the advantage of compensating higher-order motion regions with the more accurate but complicated affine modes while using the simpler translational modes elsewhere. As shown in Fig. 7, the pixel-wise MV within each affine motion block will be defined by the motion information of three corner control points as shown, by jointly considering the linear distance of the pixel position (x, y) to each control point. In this work, we accept such model as the basis of our affine motion implementation to be explained in detail more in Section 3.

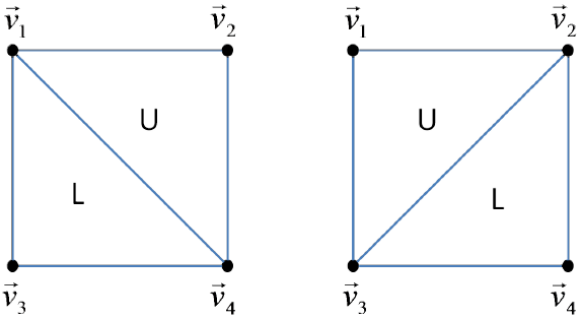

Fig. 6. Example of LU and UL mesh cells in [29]

Affine blocks will then be encoded into the quadtree based structure along with other ones. Certain R-D optimization will be utilized to select the best mode for motion compensation. However, three-control-point affine MC blocks usually requires more bytes than

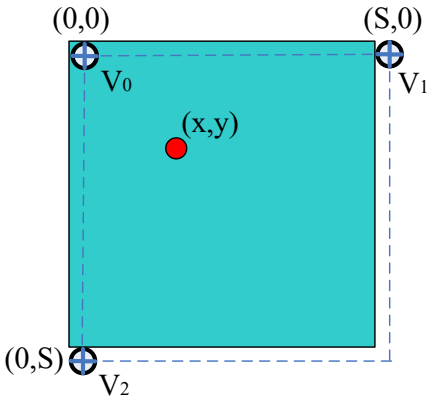

Fig. 7. Affine motion block with three control points

translational ones to transmit its higher-order motion pattern, the time cost for affine motion estimation is also considerable. Therefore, researchers then focus on topics such as improving the estimation efficiency by introducing a gradient-based method [12], increasing the option for temporal/spatial MVP candidates to minimize affine MV residual for control points [13]. Based on our work, we make the conjecture that combining affine model with the thought of state-of-the-art MCTF mechanisms such as *MERGE* may shed light on further enhancing its performance.

2.5 JPEG 2000 for Video Subband Coding

For enhanced MC-EZBC, the MCTF framework yields multiple luminance/chrominance subbands for each GOP, which will then be encoded in like manner as images by the embedded zero-block coding (EZBC) mechanism [19]. As a highly scalable video coder, an extractor ("pull" function) is utilized to obtain a scaled bitstream at a certain bitrate, thus realizing scalability. However, the current EZBC coder can only achieve one-time extraction from a originally encoded file (i.e., the code stream generated with the highest bitrate), which fits the scenario of a traditional video multicast network [14]. Later researchers proposed an advanced, hierarchical distribution schemes which allow successive extractions directly from the bitstreams stored in the caches of other subscribers [4] and thus highly improve network throughput. Such network mechanism requires the code stream to be multi-time extractable. For this to occur, the subband coder must transfer to the pulled bitstream certain control information to permit subsequent extraction.

An alternative back-end subband coder is JPEG 2000. The classic *Kakadu* software [34] which implements the main functions of JPEG 2000 allows high bitrate scalability by applying multiple quality layers. JPEG 2000 has found great success in applications such as digital cinema [16] and reconnaissance [17] as a fully scalable video coder. For encoding video subbands after MCTF processing, earlier researchers' experience showed that detailed settings also need to be modified compared with image coding settings [30]. We will find that JPEG 2000 with appropriate settings can achieve a comparable or even better PSNR performance than EZBC coder, when paired with an advanced MCTF front-end as described above.

2.6 Multiple Motion Patterns in Motion Block

In traditional block-matching, each MC block will employ a single MV set to obtain compensation value from the forward/backward reference frames. However, the relatively complicated motion within real video frames may result in multiple motion patterns within a single MC block. Fig. 8 shows an example of a region that contains two moving objects with distinct motion vectors MVp and MVq, respectively. Two moving parts are divided by a relatively clear boundary, which however can be complicated so that traditional quadtree based MV decomposition will be inefficient and costly, such as an example frame of $Mobile$ as shown in Fig. 9. Since motion compensation based on quadtree remains the primary way of video compression, trying to detect multiple motion patterns within one single block attracted many researchers' attention.

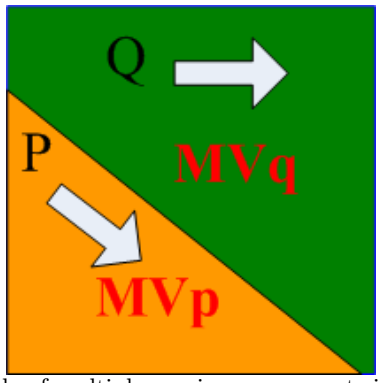

Fig. 8. Example of multiple moving components in one MC block

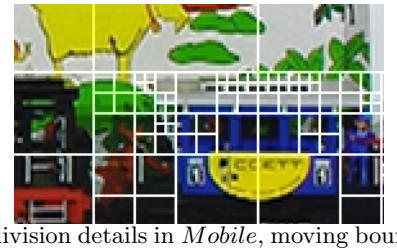

Fig. 9. quadtree based block division details in $Mobile$, moving boundaries tend to be compensated by smaller blocks

Bergen et al. [35] first proposed a three-frame algorithm to detect the two-component motion patterns among video frames, and hypothesized that both MVs are constant across the three consecutive frames. An iterative method was employed to obtain two MVs for different motion patterns, which could then be used for image/video understanding. Although limited by the requirement of temporally constant motion pattern, this work shed light upon the possibility of detecting multiple MVs within given region of a video frame. Later researchers then employed a wedge-based geometry block partitioning (WGP) method to better compensate blocks that contain multiple motion patterns [38], where a straight partition line defined by two parameters ρ and θ is encoded to describe the boundary of two motion components:

$$xcos\theta + ysin\theta + \rho = 0 \tag{5}$$

where (x, y) denotes the coordinate with respect to the upper-left corner of current block. However, a simple partition line may be insufficient to describe complicated, irregular motion boundaries, other researchers then turned to image segmentation/edge detection to help detect multiple moving components within MC block, since the motion boundaries usually overlap with spatial edges within each frame. Examples are Kim et al.'s K-means clustering method [37] and Chen et al.'s luminance threshold-based method [36]. These methods are different from wedge-based geometrical partitioning in that they make use of detected object edges to divide MC blocks into two or more components with different motion patterns.

Based on these works, Wang et al. [40] then used a Canny edge detector to further improve the performance of motion-edge detection within a block. Both luma and chroma components were taken into consideration while Canny operator worked to improve the accuracy of locating object boundaries. The motion block will thus be divided into foreground moving part to be compensated by different matching part in reference frame, which resulted in the object-boundary-based geometry partitioning (OBGP) method to implement two-component motion blocks based on H.264/AVC. Simulation results showed that coding performance of OBGP strategy outperformed both H.264/AVC anchor coder and the one with WGP methods. Such idea was then introduced into H.265/HEVC coder and discussed [41].

2.7 Adaptive Lagrange multiplier Selection for Bitrate-Scalable Video Coder

For non-scalable coders such as H.264 or H.265, the overall Lagrange multipliers are applied in the form of QP values [42] to balance the bit allocation so that the optimized decoding result with respect to certain target bitrate can be achieved. For a fully scalable video coder like enhanced MC-EZBC, however, multiple code streams with different target bitrates will be extracted from the output of encoder, thus requires $lambda_m$ as described in Section 2.1 to be adaptively revised when necessary, thus achieving relatively good performance within the range of bitrate that subscribers are most interested in.

For given total coding bits with respect to a section of a video clip (say, one GOP), more bits assigned to motion-compensation can reduce the distortion after it, then the rest of bit budget will be utilized to encode the temporal-high/temporal-low subbands. High rate quantizer theory [46] reveal that a trade-off between distortion based on motion-compensation and further subband coding can be represented as:

$$D = N^2 g \sigma_{wgm}^2 2^{-2R}, \tag{6}$$

where N^2 means the total pixels, σ_{wgm} represents the average distortion after motion compensation, and R is the average subband coding rate expressed in bits per pixel (bpp). The optimal allocation of overall bit budget into motion-compensation and subband coding highly depends on the characteristics of the videos. Chen et al. [43] proposed an adaptive method to appropriately choose the Lagrange multiplier for enhanced MC-EZBC coder with respect to different temporal levels and different video clips. These existing works inspire us to further study the Lagrange multiplier selection in scalable video coder.

3 Revised Scalable Coding System

The basis of work in this grant is our existing highly scalable video coder *enhanced MC-EZBC*. In this Section, we discuss revisions to this existing coder in order to bring it up to the state of art today. We also provide a new coder back-end which consists of JPEG 2000. The resulting new coders are termed *Interframe EZBC* and *Interframe JPEG 2000*. The Section ends with a comparison of coding efficiency gains with H.264/AVC and H.265/HEVC as references.

3.1 Revision of Translational MC Strategies

Both enhanced MC-EZBC and H.264/AVC [20] employ MV predictors for the current MC block that are medians of spatial neighbor MVs. However, Laroche et al. [11] revealed the potential performance improvement that can be obtained by selecting an optimal MV predictor, i.e. an MV prediction-competition mechanism.

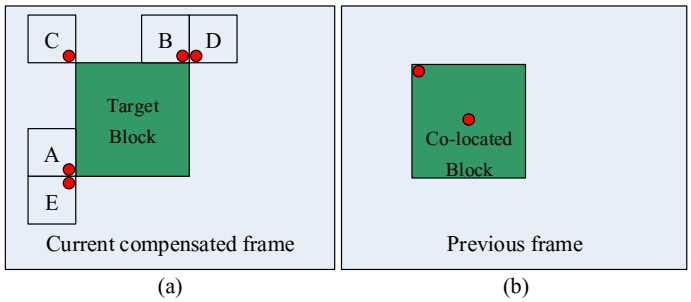

Fig. 10. Translational MV prediction candidates from (a) spatial (b) temporal neighbor positions

In our work, an MV prediction-competition is implemented as illustrated in Fig. 10. Each target block may obtain its MV predictor information from five spatial neighbor positions. Each predictor information source pixel (circle dot) may represent the MV of one neighbor block (A-E). However, some of these candidates may be unavailable due to the employed causal Z-scan order of processing the blocks, or they could provide duplicate motion information as a result of variable block sizes. We thus must appropriately manage the candidates based on their availability and create a trimmed list without any replicas. Besides spatial candidates, we also consider the MV at the same spatial position (collocated) in the previous compensated frame as a temporal MV predictor. This estimate may be obtained from the upper-left corner/center of the current block's corresponding position in the previous motion-compensated frame at the current temporal level. For the typical bidirectional MCTF mode, two candidate lists are maintained for forward/backward MC respectively. In our implementation, we create each candidate list with at most 4 candidates selected from spatial/temporal predictors with their median values as a possible supplement. If none of these candidates is available, a zero-value predictor is used. For each compensated block, we create a candidate list and then perform MV search based on the available candidates to find the one with best overall R-D cost. We thus generate MVs for the various compensation modes (forward, backward, bidirectional or other special MC modes) for the current block,

as well as a predictor index with respect to the candidate list. Then the list is checked again to select the one candidate that minimizes the bitrate cost R_{MV}. All MV residuals are then encoded by context-adaptive binary arithmetic coding (CABAC) as in enhanced MC-EZBC [15].

We also create a translational-motion candidate list with at most 4 members to implement MERGE mode for each MC block, with candidates selected from the similar spatial/temporal neighbor blocks used in the MV prediction competition mechanism. This new mode replaces the former SPATIAL DIRECT mode used in enhanced MC-EZBC [21]. Unlike the implementation [26] that decides the mutual merging relationship of blocks during an additional, non-causal round after conventional MV search, our merging method strictly maintains the causality relationship between all blocks: each block will search the candidate list for the best possible distortion cost of MERGE mode, then the estimated cost for encoding a candidate index will be jointly considered to perform the R-D cost competition. Once a block is included into a merged region, the MVs inherited from this region will also provide prediction information for other neighbor blocks according to their causality (of scanning) relationship.

3.2 Block-based Affine Motion Modes for Scalable coder

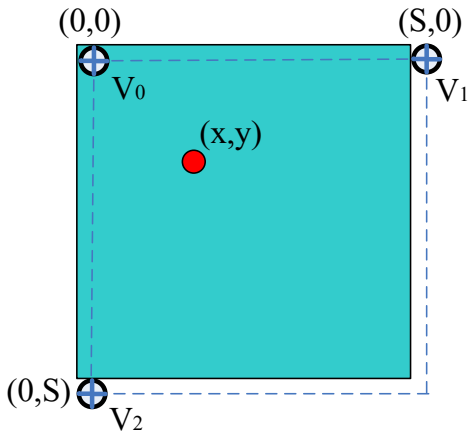

Fig. 11. Affine motion model with three control points

In our coder, the affine model with three velocity control points similar to [24] is employed to describe the pixel-wise motion of a block with a size of SxS as shown in Fig. 11. The three control point velocities (circles with blue cross) are located at the upper left, upper right, and lower left corners, respectively. Each control point has a motion vector V_i, with horizontal and vertical components given as (v_{ix}, v_{iy}). Thus the motion information of any pixel with coordinate $(x, y), 0 \leq x, y \leq S-1$ in the block can be described as:

$$\begin{aligned} v_x &= \frac{v_{1x}-v_{0x}}{S} \cdot x + \frac{v_{2x}-v_{0x}}{S} \cdot y + v_{0x} \\ v_y &= \frac{v_{1y}-v_{0y}}{S} \cdot x + \frac{v_{2y}-v_{0y}}{S} \cdot y + v_{0y} \end{aligned} \tag{7}$$

The space-variant motion vectors of an affine block are thus defined. Note that control-point velocities V_1 and V_2 are located slightly outside the block with coordinate $(S, 0)$ and $(0, S)$, respectively. Such a setting implies that the motion field defined by a three control-point affine block may also affect neighboring areas. Similar to the MV prediction-competition system of translational modes, the affine block will take the MVs of neighbors as predictors for its control-point velocities and then search for MV residuals. In our work, we consider the following sources of prediction information as shown in Fig. 12: for affine compensated target blocks, the control-point motion vector V_0 will take the motion information of the neighbors $\{A, B, C\}$ as possible spatial predictor candidates. For a system with hybrid motion models, each candidate block can be compensated by either translational or affine MVs. A translational candidate can provide a block-based, flattened MV, while the predictor provided by an affine candidate should be the space-variant MV defined by its affine motion field at the position of the target block's control-point, as described above. V_1 and V_2 also have spatial predictor candidate lists $\{D, E\}$ and $\{F, G\}$, respectively. For each control-point, the pixel-wise motion information of its collocated position in the previous MC frame will be taken as the temporal predictor. Last but not least, we also think of the selected translational MV of the target block itself with certain modes as possible predictors for the affine search. Similar to the translational MV prediction system, we trim the affine candidate lists to delete replicas and limit their lengths so that each control-point has at most 3 candidates. For a typical affine block, a predictor set $V_p = (V_{p0}, V_{p1}, V_{p2})$ can thus be obtained.

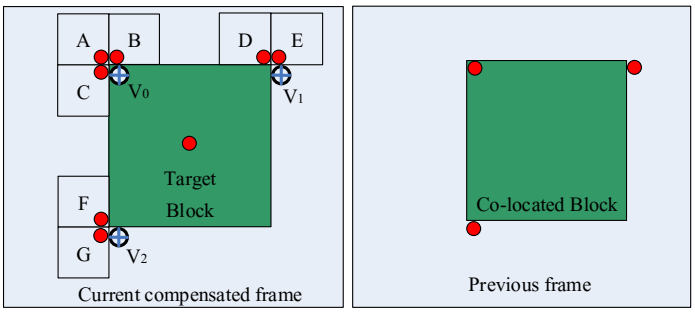

Fig. 12. Affine predictor system from (a) current frame. (b) previous motion-compensated frame

In our implementation, necessary control bits will be transmitted to indicate the affine block mode: forward, backward, or bidirectional. Then more detailed mode information further describes how affine MC is performed in the current block.

Based on the affine prediction system, we search among all available predictor sets to get the one which can achieve the lowest R-D cost. If such a cost is "optimal", then we can avoid transmitting the MV residual to reduce motion information coding bits, which constitutes the AFF-DIRECT mode. An index is transmitted to indicate the selected predictor set obtained from the various temporal/spatial nearest neighbor blocks. Based on the obtained predictor set, a local search will also be done for MV residuals of the three control-point velocities, which may further reduce the R-D cost. Then additional MV residuals of the control-point velocities will be transmitted, we call it AFF-INTER mode. Inspired by the

PARALLEL mode of enhanced MC-EZBC for translational MC blocks [20] which makes use of the consecutiveness of motion and may reduce half of the bit cost for encoding MV residuals, an AFF-PARALLEL mode is applied to bi-directional compensation blocks, where the forward affine MVs will be searched in a similar way as AFF-INTER and the backward MVs are inferred from the inverse of their opposite counterparts.

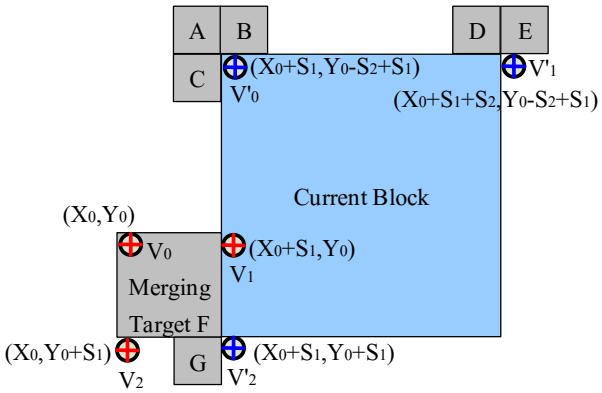

Fig. 13. Motion-estimation process for the new coder

Similar to the translational block merging mechanism, the three control-points of an affine MC block can also "inherit" motion information from certain near neighbor blocks, i.e. by being merged into the motion field of such a neighbor, thus the motion of a region consisting of multiple blocks may be represented with a continuous affine motion field. This kind of AFF-MERGE mode can be implemented in a way as shown in Fig. 13. For the current block with a size of $S_2 \times S_2$, spatial neighbors A-G are possible merging targets. For instance, if neighbor block F with a size of $S_1 \times S_1$ is an affine compensated one with control-point motion vectors $V_i = (v_{ix}, v_{iy})$, then the current block can be merged into this neighbor through deriving its control point motion vector $V_i' = (v_{ix}', v_{iy}')$ according to (2) and the coordinates of each control-point:

$$
\begin{aligned}
v_{0x}' &= v_{1x} + \tfrac{(v_{2x}-v_{0x})\cdot(S_1-S_2)}{S_1} \\
v_{0y}' &= v_{1y} + \tfrac{(v_{2y}-v_{0y})\cdot(S_1-S_2)}{S_1} \\
v_{1x}' &= v_{1x} + \tfrac{(v_{2x}-v_{0x})\cdot(S_1-S_2)}{S_1} + (v_{1x}-v_{0x})\cdot S_2/S_1 \\
v_{1y}' &= v_{1y} + \tfrac{(v_{2y}-v_{0y})\cdot(S_1-S_2)}{S_1} + (v_{1y}-v_{0y})\cdot S_2/S_1 \\
v_{2x}' &= v_{1x} + (v_{2x}-v_{0x}) \\
v_{2y}' &= v_{1y} + (v_{2y}-v_{0y})
\end{aligned}
\tag{8}
$$

However, in some cases the MV of the most "remote" control-point (which is V_1' in this example) cannot be well described by the motion field of the merging target, especially when the size of the current block is relatively large. Therefore, we also design a "partial" merge mode if the merging target is B, C, D, or F as shown in Fig. 13, where the motion of the two

control-points closer to the merging target (which are V_0' and V_2' in this example) are derived according to the method as mentioned above, while local search similar to *AFF_INTER* can be performed on the last control-point to minimize the overall R-D cost of the current block. Moreover, such a mode can even take translational neighbors as possible merging targets by regarding their motion as a special case of affine motion where all control-point MVs are identical, thus allowing a merged region to spread across both translational and affine motion fields. Conversely, for the "full" merge cases that all control-points are incorporated, no additional motion information other than merging target index needs to be transmitted any more. AFF-MERGE mode can significantly reduce affine motion information coding bits like AFF-DIRECT, however, the former one achieves that by creating a continuous affine motion field. Of course, there will also be a concomitant reduction in blocking artifacts.

3.3 Motion Estimation Processes

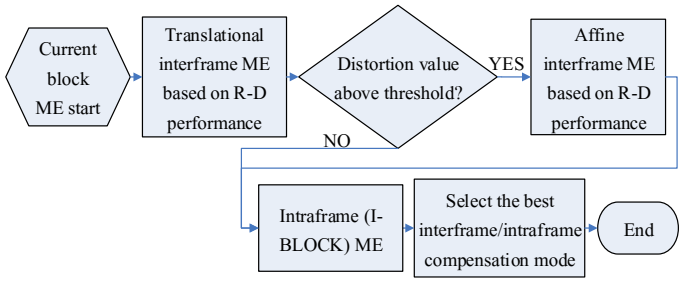

Fig. 14. Diagram of motion estimation (ME) algorithm for the new coder

As shown in Fig. 14, we will try the interframe translational modes for each block first and select a relatively "best" mode according to the R-D tradeoff. If the distortion measured in mean absolute difference (MAD) is still above a certain threshold (more than 1 in our experiments), then the more time-consuming affine search will be done. Next, enhanced MC-EZBC will also try intraframe (I-BLOCK) compensation modes for blocks with relatively small sizes (say, 4×4 or 8×8 blocks), which is inherited in our coder. Then all interframe/intraframe MC modes applied to the current block will go through R-D competition and finally select the best one.

3.4 Subband/Wavelet Coding of Spatiotemporal Residuals

After MCTF, the generated luminance/chrominance temporal subbands will be encoded by the EZBC coder with a maximum bitrate. The "pull" function of enhanced MC-EZBC can then extract code-streams with target bitrates to achieve "bitrate scalability". However, the EZBC codec currently only allows a one-time extraction from the originally encoded bit file, which may not fit a hierarchical distribution scheme [3]. Also, we here introduce JPEG 2000 as a temporal subband coder backend for our new MCTF, which has found great success in digital cinema [16] and reconnaisance [17].

In our JPEG 2000 implementation, for each GOP and after the MCTF consisting of 3 temporal-low subbands (Y/U/V one each) and multiple temporal-high ones, JPEG 2000

Table 2. Motion compensation modes for Interframe EZBC

Mode No.	Block mode	MCTF direction	Update step	MV residual
1	bi-connected	BI	Yes	2
2	left-connected	LEFT	Yes	1
3	right-connected	RIGHT	Yes	1
4	bi-predicted	BI	No	2
5	left-predicted	LEFT	No	1
6	right-predicted	RIGHT	No	1
7	parallel	BI	Yes	1
8	block-merging	TBD	Yes	0
9	I-block	INTRA	No	0
10	bi-affine	BI	Yes	2
11	left-affine	LEFT	Yes	1
12	right-affine	RIGHT	YES	1

(we use Kakadu [33]) will compress all of them into one bitstream by employing multi-component settings. In effect, each temporal subband is regarded as a "component" as defined in JPEG 2000. Multiple quality layers can be set to obtain the desired bitrate scalability. Through appropriately selecting parameters such as levels for discrete wavelet transform (DWT) and quantization steps for these temporal-high/low subbands, JPEG 2000 can achieve similar performance to EZBC. Further, using JPEG 2000 quality layers, subsequent bitrate reductions will then be possible for hierarchical video distribution.

3.5 Experimental Results and Discussion

This section will present experimental results with the incorporated new improvements mentioned above. The first part will concentrate on results for the new Interframe EZBC. This is followe by a section on comparisons including the new Interframe JPEG 2000. The comparisons also include results for the nonscalable standards-based coders H.264/AVC and H.265/HEVC.

Improvements Obtained by New MCTF Framework Based on EZBC subband coder, the modification of MC modes in enhanced MC-EZBC are as shown in Table 2: three new modes for affine motion-compensation have been added, all of them require the prediction error to be low enough for the update step so that we can make sure higher-order motion model really brings the motion prediction error down to an ideal level. Within each category of affine mode, more indices will be transmitted to further indicate the MV predictor, existence of MV residual, parallel backward MV and so on. The SPATIAL-DIRECT mode for translational motion blocks has been replaced by a BLOCK-MERGING mode, which incorporates affine-motion cases as described in previous section.

We tested multiple video clips to show the possible advantages of our advanced MCTF framework. Results are obtained by three different coders: 1) original enhanced MC-EZBC coder; 2) EZBC subband codec with translational MV prediction competition/Block Merging; 3) both translational and affine revisions. The quadtree block size ranges from 4×4 to 64×64. To save encoding time cost, we only employ affine motion estimation (ME) in blocks no smaller than 8×8. The motion-search range of temporal level 1 (as described in section I) is $[-16, +16]$, which will be doubled as temporal level goes up (i.e., frame rate goes down) until an upper limit of $[-64, +64]$ is reached at certain level. A GOP size of 32 (i.e., 5 temporal levels) is applied on all clips to achieve best compression efficiency.

Table 3. BD-rate Saving Achieved by Revised MCTF Frameworks Compared with enhanced MC-EZBC

Video name	Resolution & Total frame	Coder 2	Coder 3
Bus	352x288, 150	6.15%	8.46%
Flower	352x288, 250	6.65%	12.61%
Mobile	352x288, 300	6.25%	8.81%
Tempete	352x288, 260	3.11%	9.25%
Flowervase	832x480, 300	7.90%	12.95%
Big Ships	720x480, 300	4.32%	17.03%

Table 3 lists the video clips employed in these experiments (all 30 frames/s). We use three coders as described above to compress these video clips and calculate the BD-rate saving that can be achieved by advanced MCTF frameworks compared with enhanced MC-EZBC. The bitrate savings measured in terms of luminance (Y) BD-rate savings according to [31] are given. Results show that all clips benefit from the revised MCTF framework to various degrees. Among these clips, coder 2 with translational MV competition and block merging achieves the most bitrate saving of 7.9% when coding *Flowervase*, while coder 3 with all translational/affine MCTF revisions achieves both the highest overall bitrate saving of 17.03% compared with original enhanced MC-EZBC and the most significant bitrate saving by introducing the affine motion model compared with coder 2 in processing *Big Ships*.

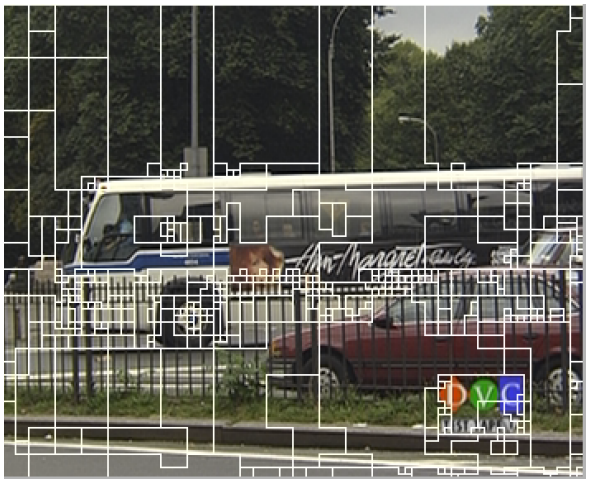

Fig. 15. Merged regions in frame 118, temporal level 1 of *Bus* obtained by coder 2

Fig. 15 shows the translational merging areas (divided by white lines) of frame 118, temporal level 1 of *Bus* obtained by coder 2, each area consists of multiple compensation blocks that share identical motion information. The panning of the camera makes it difficult to describe the motion of larger areas with a single translational MV, but we can still observe that some typical objects (such as the two streetlight poles) are within the same merged regions.

For videos with typical camera zoom-in/zoom-out, the motion between frames at higher temporal levels (lower frame rate) are difficult to describe with only translational modes. The left picture of Fig. 16 shows the MC blocks/quadtree structure within frame 22, tem-

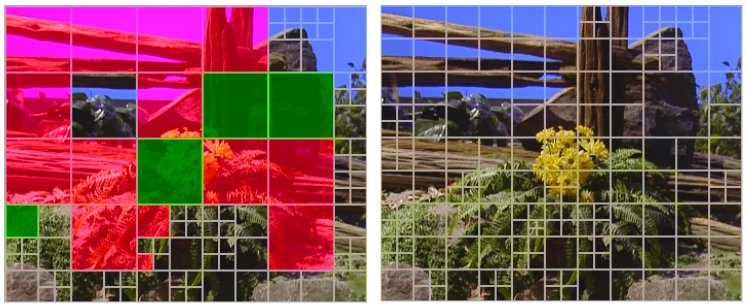

Fig. 16. Change of block division by introducing affine modes (left) compared with translational-only case (right), where affine merge blocks (green) and other affine blocks (red) were highlighted

poral level 3 of *Tempete* obtained by coder 3, where affine merge blocks (including "partial" and "full" merging ones) and other affine blocks are marked with green and red colors, respectively. Compared with results of the same frame obtained by coder 2 with only translational modes, significantly larger blocks replace complicated quadtree structures consisting of small blocks in most affine-motion areas. An affine block usually needs more bits than its translational counterpart to encode its higher-order motion information. However, the highly simplified quadtree structure with lower overall number of blocks along with appropriately employed merge modes can effectively neutralize such impact. In fact, we find that video clips processed by coder 3 usually require similar or even less total bitrate than coder 2 to transmit the motion data (including MC modes, predictor and MV residual information). For processing the two SD size videos *Big Ships* and *Flowervase*, the total motion data generated by coder 3 are 85.5% and 98.3% those of coder 2, respectively. Such an outcome also helps enhance performance at lower bitrates, where a significantly higher proportion of bits are used to encode motion data.

Comparison Between Different Scalable Subband Coders Here we compare the performance of the new Interframe EZBC and JPEG 2000 coders under the MCTF framework of coder 3 as described in the last subsection. For JPEG 2000, the Kakadu *v7.8* software [33] is employed to process the temporal subbands. Multiple quality layers as described in [34] are compressed into the code-stream file to achieve scalability, where a new quality layer is set for each total bitrate increase of a certain amount (1000 Kbps for HD video and 150 Kbps for others). Each quality layer contains a certain amount of subband data and will be discarded if no bytes from such layer are extracted into the decoding bitstream. For each GOP, 5 levels of discrete wavelet transform (DWT) are performed on the temporal-low subband which has characteristics of a typical low-pass image, while other temporal-high subbands will go through only one level of DWT. However, for clips such as *Bus* or *Big Ships* which contain relatively fast motion, three levels of DWT should also be performed on the spatial high-frequency subbands of the highest temporal level to enhance coding efficiency since they contain more lowpass features due to the errors of motion compensation.

The H.264/AVC and H.265/HEVC coders with high profile settings are also important references, so here we configure the simulation environment mainly according to [32] and choose the "I-B-B-P" frame structure. An Intra-frame period (GOP) of about 30, the maximum motion estimation search range of $[-64, +64]$ and *Fast full search* mode are used, which are comparable to the basic settings of our scalable coder as mentioned above. Results of some typical clips are as shown in Fig. 18 and 19. For the 1920×1080 HD video

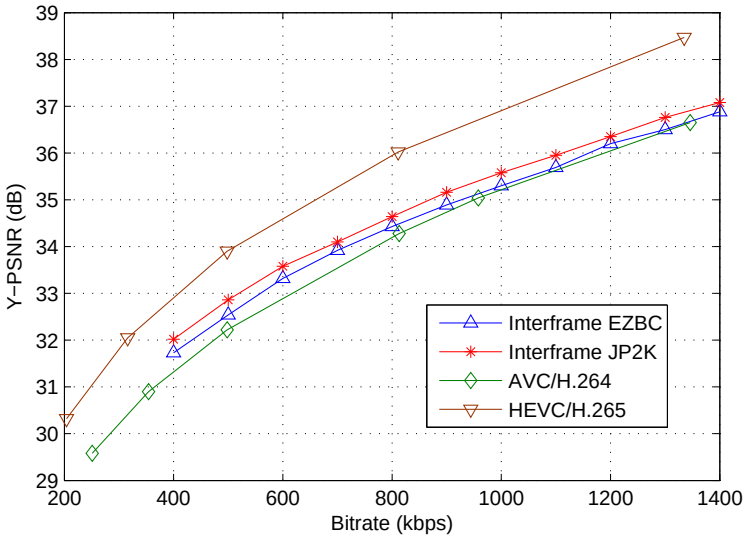

Fig. 17. Y-PSNR results of *Tempete*, obtained by different coders

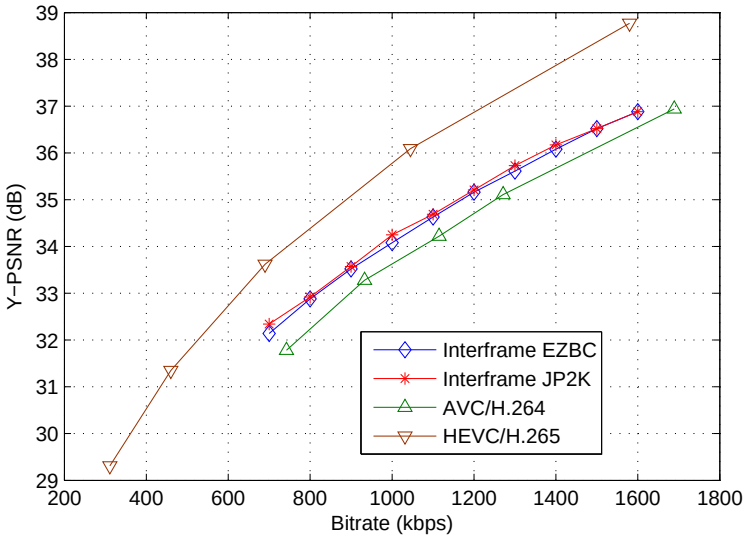

Fig. 18. Y-PSNR results of *Flower*, obtained by different coders

BQTerrace (60 frames/s, we test on the first 240 frames only), its chroma resolution is not a multiple of 8 and cannot be processed by EZBC due to an existing software limitation, while the JPEG 2000 Kakadu coder can easily handle it. We can find that the coding performance of EZBC and JP2K subband coders are comparable, for *Tempete* and *Flower*, JP2K even performs slightly better than EZBC. Further, with the help of multiple quality layers, the JP2K coder can achieve multi-level extraction while achieving a similar PSNR performance to that of the EZBC coder. Compared with H.264/AVC, our Interframe EZBC and Interframe JP2K scalable coders also achieve significantly better Y-PSNR performance.

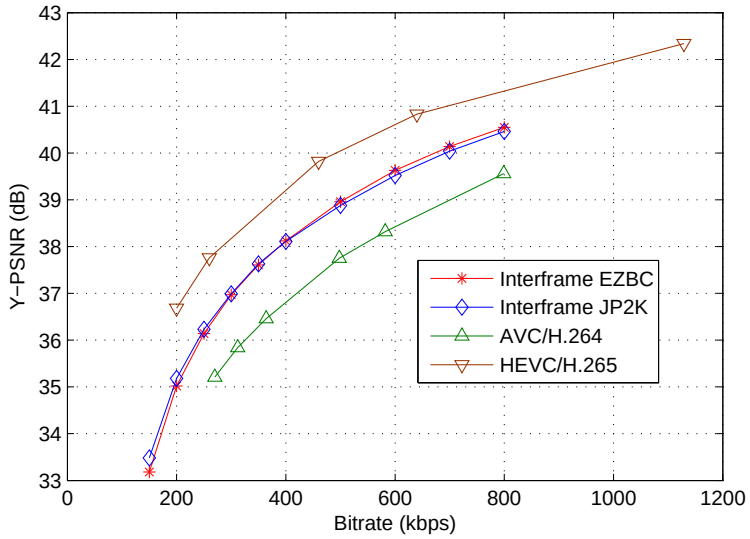

Fig. 19. Y-PSNR results of *Flower Vase*, obtained by different coders

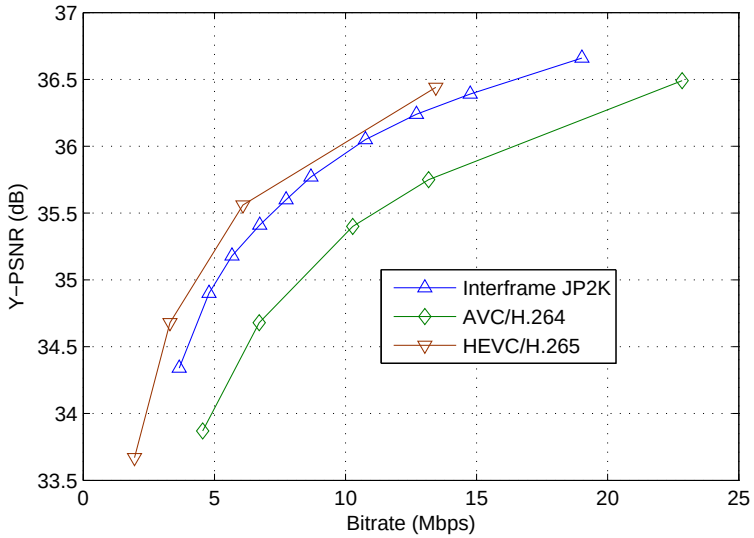

Fig. 20. Y-PSNR results of *BQTerrace*, obtained by different coders

In conmparison against H.265/HEVC, performance is still a bit low except for the HD clip BQTerrace. Ongoing research is following up this matter in the completion of the doctoral thesis of Mr. Yuan Liu, which has been supported up to 30 August 2018 by this AFOSR grant. Completion of Mr. Liu's doctoral thesis is expected in June 2019.

4 Two-component Motion Compensation Mechanism for Block-based MCTF

Here we present in detail our work in this grant on the topic of two-component motion in a single block, a new technique in the scalable video coding. This work was not finished in the grant, but continues on in the doctoral thesis of Yuan Liu expected to be completed in June 2019. Current status is that he is just beginning to get good results on real test clips. This Section, presenting results as of 30 August 2018 only shows coding results for a synthetic motion created in an original animation test clip.

4.1 Three-Frame based Two-Component Motion Detection

In traditional block-matching based motion compensation strategies, each MC block in compensated frame will employ a specific MV set to obtain compensation value from the forward/backward reference frames. However, the motion within real video frames may result in multiple motion patterns within the region of a MC block. Fig. 8 shows a typical example of a region that contains two moving objects with distinct motion vectors MVp and MVq, respectively. In real motion estimation, two moving parts are likely to be divided by a relatively clear yet irregular motion boundary, which makes traditional quadtree block based MC inefficient and costly.

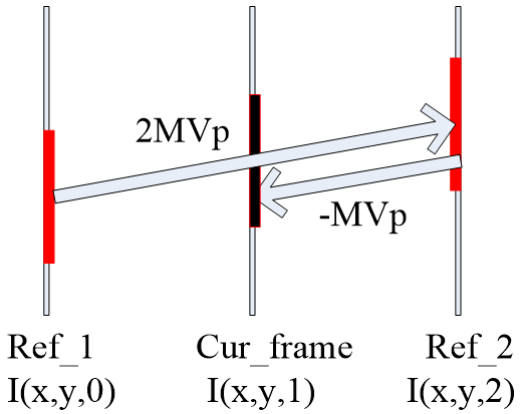

Fig. 21. Three-frame model for obtaining two-component MVs

The development of HD videos result in increasing frame rate, thus also reducing the temporal distance between compensated/reference frames and making the motion between several consecutive frames more likely to be constant. Based on the temporal decomposition structure of each GOP in ENH-MC-EZBC codec which will provide each compensated frame with two reference frames on forward/backward sides, we try to employ Bergen et al.'s two-component method [35] as follow:

Within each temporal level of ENH-MC-EZBC, the pixels within a two-component motion region are sorted into clusters **P** and **Q** (i.e., the two distinct components) as shown in Fig. 8, which have motion vector **MVp** and **MVq**, respectively. Such motions can shift the position of each component consecutively. We denote them in the form of $I(x, y, t)$, where (x, y) represents the spatial position and t refers to time.

Then the difference blocks **D1** and **D2** between corresponding positions of **Ref1/Ref2** and **current frame/Ref2** can be denoted as:

$$\begin{aligned} D_1 &= I(x,y,2) - I^{2p}(x,y,0) = (P^{2p} + Q^{2q}) - (P^{2p} + Q^{2p}) = (Q^{2q} - Q^{2p}) \\ D_2 &= I(x,y,2) - I^{p}(x,y,1) = (P^{2p} + Q^{2q}) - (P^{2p} + Q^{p+q}) = (Q^{2q} - Q^{p+q}) \end{aligned} \tag{9}$$

Thus we can obtain:

$$D_1 - D_2 = (Q^q - Q^p)^p = D_2^{(p-q)} \tag{10}$$

Then we can calculate a second-level difference block between **(D1 - D2)** and **D2** to find a proper value of **(p-q)**, and **MVq** can be automatically inferred from **(MVp-MVq)** and known **MVp**.

During motion estimation, the **D1** and **D2** as described in last section will be generated by taking the one-component MV obtained by current target block as the initial **MVp**, which have the same size as the residual block we want to encode for two-component block. Then we calculate **(D1 - D2)** and "shift" its position in the way as shown in Fig. 22:

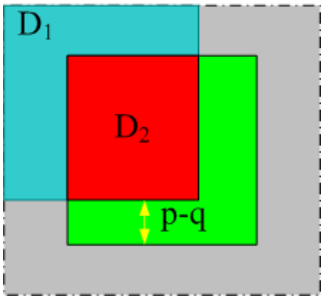

Fig. 22. Generation of Second-level difference block

By shifting the position of **D2** around, we may find a position that part of **(D1-D2)** and **D2** can perfectly match each other (red area). Then we obtain the value of **(MVp-MVq)** and can then calculate **MVq**. Then we reselect **MVp** by fixing the value of **MVq**. Several rounds of such searching process based on sum of absolute difference (SAD) result in two-component MVs.

Note that such Bergen et al.'s method was initially only used for video feature understanding, which obtains two motion components/MV sets without knowing exactly which MV set is used to compensate each part of the current two-component block. However, in real motion compensation, we need to know this on both encoding and decoding sides to do effective MC analysis/synthesis.

Thus we rely on Canny edge detection algorithm for two-component MCTF, the following section describes in detail how Bergen et al.'s two-component MV detection method is effectively combined with Canny edge detection in video compression. We should also note that Bergen method only covers a special case of two-component motion (i.e., constant motion among three temporally consecutive frames), other methods (eg. Wang et al.'s [40])

can also be used to obtain MV sets for two-component motion blocks, which will also be encoded by using Canny edge detection. The rest of this section briefly shows how Canny operator can effectively perform two-component MCTF with the obtained two-component MV sets.

4.2 Two-component MCTF Implementation Based on Canny Edge Detection

Different from Bergen et al.'s method which avoided searching for the boundaries of different motion components, later researchers turned to making use of geometrical partitioning or edge detection methods in implementing multiple motion components within an MC block. By either geometrically approaching motion boundaries or detecting them from the luminance/chrominance information of frames, a motion block will be further divided into at least two parts and find their respective compensation. Limited amount of side information will thus be transmitted to help reconstruct such block on decoding side.

Motion Boundary Detection based on Canny Operator Wang et al. [40] highly summarized these early attempts. As a result, a method based on canny edge detector was proposed to do the job. A standard canny edge detection algorithm consists of the following steps:

1) Use a Gaussian filter to eliminate the noise and prevent false detection. This step smooths the image to reduce the effects of obvious noise on the edge detector. A parameter σ is used to determine the size and coefficients of smoothing operation;

2) The intensity gradient of video frame will then be found by four filters, thus possible horizontal, vertical and diagonal edges in the blurred image can be detected;

3) A non-Maximum suppression is applied to "thin" the edge, for the edge indicated by gradient value is still quite blurred while an only, accurate response to the edge is necessary for two-component motion estimation. Thus non-maximum suppression can help to suppress all the gradient values by setting them to 0 and only preserve the local maxima, which indicate locations with the sharpest change of intensity value and will be marked as edge pixels;

4) Two threshold parameters t_{low} and t_{high} will be utilized to further filter edge pixels, those with lower gradient values lower than t_{low} will be removed while those between t_{low} and t_{high} will be marked as "weak" edges;

5) A hysteresis mechanism will finally extract edge information by first accept "strong" edge pixels with gradient values higher than t_{high} and then double-check all "weak" edge pixels by only keeping those which have at least one "strong" edge pixel out of its 8-connected neighborhood pixels.

We added necessary algorithms into the Interframe EZBC codec to implement Canny edge detection. Figure. 23 shows the edge detection result of the first frame in *Foreman* clip, we can see that most outlines of the human (which tend to be motion boundaries of two-component mode) was detected.

Further post-processing is still needed to close possible tiny gaps on each edge, then pixels divided by existing edges will be sorted into multiple clusters. In order to reconstruct the identical information for current motion block on both encoder and decoder's sides, the Canny detection will be performed on reference frame so that the foreground/background motion components can find their best match in reference frame.

Fig. 24 gives a typical example in the frame 2 of *Tennis* clip, the block which contains a fast-moving ball cannot find a single good match in reference frame, which results in

Fig. 23. Object edges obtained by Canny operator with $\sigma = 0.8$, $t_{high} = 0.7$ and $t_{low} = 0.2$

a lot of small quadtree motion blocks as shown. However, when we do a full search to locate the motion boundary (i.e., the outline of the ball) as well as the foreground moving object (the ball itself), then a background-compensated prediction (BCP) as mentioned in paper [40] will be performed on the rest part of the block to compensate the background component. Motion estimation result shows that when encoded as a single MC block, Canny method reduced 64% of the SAD in such 32×32 region compared with the best outcome of traditional one-component MC modes for such macroblock. Fig. 25 further shows the direct results of Canny edge detection, where foreground/background components and edge pixels are differentially marked. Edge pixels will then be classified into the two components by their luminance similarity to each part like in [40]. We can see that these steps can be completely repeated in decoder, since the motion boundaries are detected in reference frames which are supposed to be reconstructed logically before the synthesis of current frame (block).

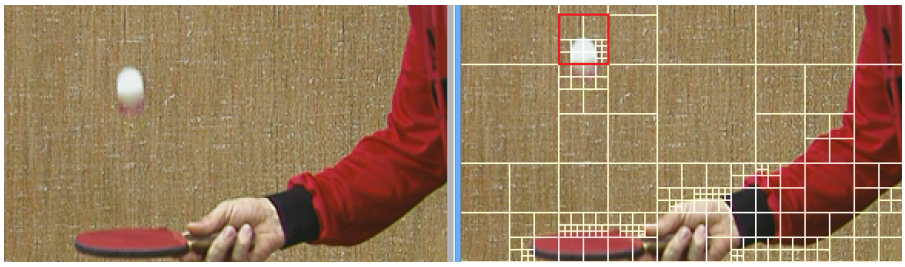

Fig. 24. Example of a typical two-component block (red) in frame 2 (right picture) of *Tennis* clip, whose foreground and background parts can be respectively compensated in forward reference frame (left picture)

Note that in this case two-component MV sets are obtained from motion estimation based on the detected Canny edge in reference frame, but MVs generated by Bergen et al.'s method or any other two-component motion estimation mechanism can rely on Canny operator to finish the MCTF processes. Our next subsection will thus show complete MCTF and coding results obtained by combining Bergen et al.'s motion estimation method with Canny edge detector, some new thinking about two-component MV implementation will be discussed and preliminarily verified.

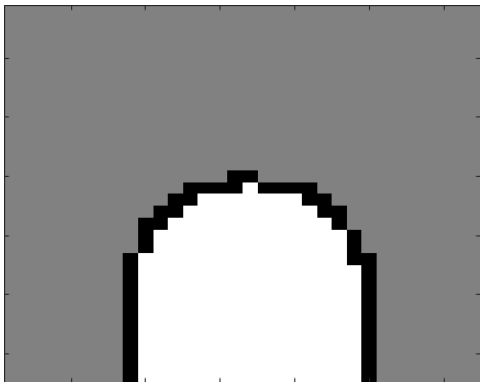

Fig. 25. Canny edge detection result of the 32×32 block in *Tennis*, foreground (white), background (gray) and edge pixels are respectively marked

Initial MCTF Results based on Canny Edge Detection By employing Bergen et al. method, now two MV sets **mv1** and **mv2** are selected to compensate the current block's different motion components. Generally speaking, motion components can be divided into foreground and background ones. Foreground component is spatially closer to camera so that no "occluded" parts of such component exist among current and reference frames, which means such component may be well compensated forwardly/backwardly/bi-directionally. As to the background component, however, it is likely that some pixels will be occluded in at least one reference picture. By utilizing Canny edge detection in both reference pictures, we can sort out foreground/background MV sets when doing motion estimation and let the decoder know it.

Fig. 26. An example of foreground/background components in Foreman

Fig. 26 shows a typical example of foreground/background components in foreman clip: The white helmet is a foreground moving part that could be effectively traced in both forward and backward reference pictures without occluded pixels, while the background (cement wall) component in current block (the middle picture) can only be well compensated by forward (left) reference picture.

Similar to Wang et al.'s two-component MC mechanism, the Canny edge for motion compensation will be obtained from reference picture so that an edge map identical to encoding side can be recovered in decoder. We use a synthesized video to test the effectiveness of such Bergen-Canny two-component motion model, as shown in Fig. 27: a foreground object (white bird) is moving toward a certain direction with constant velocity, while the background is fixed and has distinctive texture so that Canny operator can easily detect the motion edge.

Fig. 27. Synthesized Video for test usage

By letting sigma = 2.2, low threshold = 0.2 and high threshold = 0.8 in Canny operator, we obtain an edge map in the shifted block position of reference frame. Since motion edge always moves along with foreground component, so the shifted position will be obtained by foreground MV set.

For all motion edges detected for a two-component MC block, Wang et al. only chose the one that could divide current block into two parts with most distinct luminance feature to perform two-component. Such algorithm works well for many cases, but could also fail when the block size is relatively larger and relatively complicated texture exists. Fig. 28 shows an example in foreman clip: in the highlighted block, the green curve is the real edge while the yellow curve will be selected by using such method.

Fig. 28. Example of failed edge detection

Thus we find an optional way to select Canny edge when bi-directional MV sets for both components are obtained (this is true for Bergen two-component blocks, where both foreground/background components can be compensated by PARALLEL modes as employed in

traditional ENH-MC-EZBC codec): an edge map is obtained in the reference picture with "occluded" background parts (which is the backward/right reference picture as shown in the case of Fig. 28), then we can see all pixels in this reference block are not "occluded" in another reference picture. Since information of both reference pictures will be reconstructed before synthesizing current frame, thus all the components divided by Canny edges in "occluded" reference picture will be compared with counterparts in another reference picture obtained by shifting them with these two MV sets, we can confirm which parts belong to the foreground component, while all the rest parts are background ones, the only motion edge could then be selected. By creating identical two-component information on both encoding and decoding side, we can finish the MCTF processes.

Besides the intelligent edge selection algorithm, this new version of coder contains two additional mechanisms to improve the performance:

1) We realized that the Canny detector always marks the pixels where the luma/chroma value significantly changes as the edge. However, the pixels between moving object and the background may form a wider, blurred belt that includes the selected Canny edge, which we refer to as the "fringe" pixels. Fig. 29 showed an example of fringe pixels in the synthesized video frame, where the red curve represents the edge marked by Canny detector. For the synthesized video with a brighter foreground component and a relatively darker background part, we found that fringe was usually located outside Canny edge.

Fig. 29. Detected Canny edge and fringe pixels

Thus for such synthesized video clip, we designed multiple fringe types to sort out fringe pixels: based on the detected Canny edge, an index will be transmitted to indicate whether pixels within certain range (probably 0-2 pixels) of the background side of the Canny edge should be classified as foreground part. An index will then be transmitted along with mode information to indicate such type.

2) For the background part, some "occluded" pixels are very likely to exist so it should be uni-directionally compensated by one of the reference frames, while the foreground part may benefit most from certain motion-compensation mode, which can be either uni-directional or bi-directional, thus the MC modes and direction of foreground/background components should be freely, separately selected to further improve the SAD performance.

The detailed encoding processes of Bergen-Canny method to implement two-component motion compensation are shown below:

Algorithm 1 Bergen-Canny two-component MC mode

1) Two-component modes based on three consecutive frames (i.e., current frame and its two reference frames) is used to obtain two MV sets for current block;
2) Canny operator is used to detect edge map in reference frame with the shifting of foreground motion;
3) Use intelligent edge detection algorithm to select the real motion edge;
4) Select appropriate MC modes for foreground/background components, respectively;
5) Process the "fringe" pixels, transmit an index to indicate related mode;
6) Obtain difference block for subband coding, transmit all MV set and mode information.

Two-component MC mode is applied to temporal level 1-3 of Interframe EZBC codec to compress such synthesized video. Here we compare two codecs with multi-component modes with traditional block-based codec: **codec 1** employs two-component modes without introducing the three mechanisms as mentioned above, while **codec 2** contains these mechanisms.

Compared with the control group with only traditional MC modes, 46% of motion bytes are reduced by **codec 2**, while **codec 1** only reduced 18% compared with traditional codec. The details of PSNR versus bitrate curves obtained by multiple codecs are shown as in Fig. 30 and Fig. 31.

Compared with traditional codec, the **codec 1** only achieved limited gain within low-bitrate range, while **codec 2** can achieve advantage (i.e., BD-rate saving) over one-component control group when extracted bitrate ranges from 400-3000 kbps (which yields Y-PSNR values between 28-39 dB), while the previous one's performance becomes much worse when bitrate is relatively high. Within the bitrate range of 400-1500 kbps, the two-comp coder achieves 4.19% of BD-rate saving (obtained by the method mentioned in [31]), which decreases to 1.86% within 1600-3000 kbps.

5 Adaptive Lagrange Multiplier for Interframe EZBC and JPEG 2000

This Section presents a new rate control strategy for Interframe EZBC and Interframe JPEG 2000. The rate control in these highly scalable coders is accomplished by varying the Lagrange parameter $lambda_m$ in the mode decisions. This is somewhat analogous to conventional rate control in H.264/AVC which is handled by selection of the step-size parameter QP. In a research coder, typically the choice of these parameters is made by trial-and-error for each test clip. In a real or practical coder though, there has to be some method to actually control the bitrate to achieve approximation to a specified rate constraint, be it constant bitrate or variable bitrate.

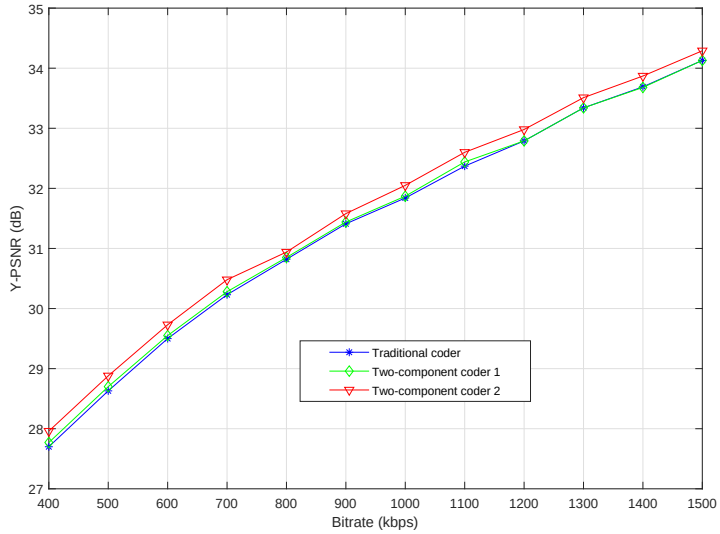

Fig. 30. Y-PSNR curves for 400-1500 kbps bitrate range

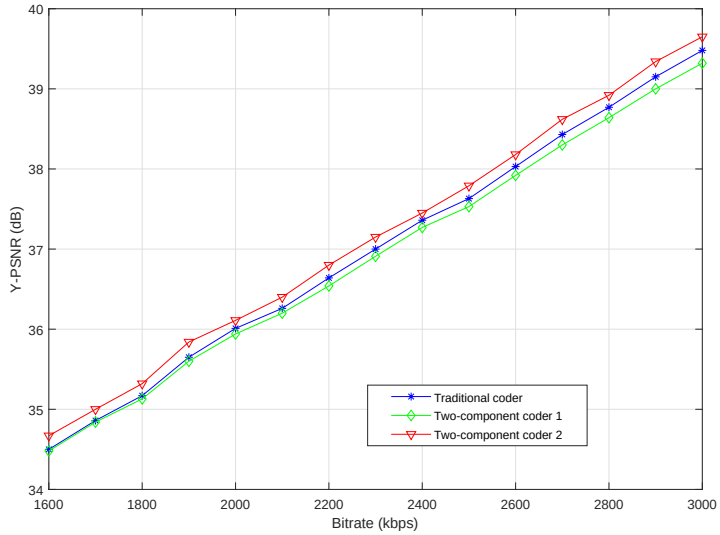

Fig. 31. Y-PSNR curves for 1600-3000 kbps bitrate range

5.1 Some Findings and Thoughts about Motion/Subband data Compression

With the fully scalable Interframe EZBC structure, MCTF decomposition will be performed on each temporal level to generate temporal-high subbands. An obvious trade-off exists between bits for coding the compensation residual of temporal-high subbands and motion information, which will vary for video segments with different motion features. Fig. 32 shows

Table 4. Comparison of Y-PSNR obtained with different groups of Lagrange multipliers

Segment No.	**PNSR obtained with λ_1**	**PNSR obtained with λ_2**
1	40.10	40.01
2	37.62	37.89

two frames from different segments of *Foreman* video clip. Frame 1-9 mainly contains a human face making theatrical expressions and a background with relatively simple texture, while the frame 282-290 contains nearly static scenes with only limited camera shake. Smaller motion λ values may be fit for scenes with more significant, complicated motion to sufficiently improve the coding efficiency and vice versa, which is also reported by other researchers [44]. Last but not least, based on the current closed GOP setting in Interframe EZBC, the complexity of coding temporal-low subbands will also more or less impact such balance.

Fig. 32. Comparison of frame 1, 9, 282 and 290 in *Foreman* clip

Thus frequent changing of these factors will jointly affect the real-time appropriate λ for motion estimation, the target bitrate will also have impact on it. Here we apply two groups of lambda values: λ_1: $(7, 8, 8)$ and λ_2: $(14, 14, 15)$ for three MCTF levels, two segments of *Foreman* clip: frame $1 - 64$ and frame $249 - 280$ are encoded with these two groups of Lagrange multipliers and then decoded with the same target bitrate of 500 *kbps*.

We can see that segment 1 slightly prefers the smaller lambda values to better trace and encode its motion, while segment 2 can obtain more significant PSNR improvement with larger lambda values. Thus when video feature significantly change, the coding performance

may sufficiently benefit from reselecting Lagrange multipliers and obtain corresponding P-SNR improvement when the feature of video frames significantly changes.

5.2 Steepest Descent Mechanism for Implementing Adaptive Lagrange multiplier

Although some features as observed in last section can help reveal the possible trend of optimizing Lagrange multipliers, we still count on a more basic way to directly verify how modifying λ value for each MCTF level will impact the performance of video coding.

In the traditional ENH-MC-EZBC framework, the encoder (pre-coder), code stream extractor and decoder work separately and non-simultaneously. However, now we sufficiently redesign the processes of encoder to let it incorporate the functions of extractor and decoder: a target bitrate (which is most likely the central bitrate for possible range of scalable extraction) will be chosen to extract corresponding code stream for each GOP from the outputted data of encoding processes. When the encoder reaches some point where the video features change significantly, the decoding functions can decompress necessary video segments and calculate the Y-PSNR value to check the coding performance obtained by different λs. If another λ value group is proven to be optimal, some pre-buffered information will allow the encoder to "rewind" back to the previous step and redo the motion estimation and MCTF analysis with such lambda values. Such mechanism can thus support a steepest descent method to implement motion λ adaptation. Some details of such algorithm is shown below:

Algorithm 2 Motion λ adaptation to optimize Y-PSNR

```
Start with motion λ group best for the first part of video
for GOP = 1 to n do
  round = 1
  while (round <= k) and (λ group modified during current round) do
    for Temporal level = 1 to m do
      Modify λ of current level within the range of (λ_cur − δ,λ_cur + δ);
      Perform motion estimation and compress current GOP;
      Simulatively decode current GOP with target bitrate, calculate Y-PSNR;
      Reselect motion λ for current level, which maximizes the PSNR;
    end for
  end while
  round = round + 1;
  Do actual motion estimation with searched motion λ group;
  Save the modified λ group as the starting point of next GOP's adaptation;
end for
```

We select the *Foreman* clip to verify the effectiveness of our adaptive λ algorithm. As one of the typical example with frequent feature change, the first half of this video mainly consists of exaggerated human expressions and limited camera jittering, while the second half contains more human gesture and camera rotation, then the camera stops at a static scene. For the two experimental codecs with/without Lagrange multiplier adaptive mechanism, an initial group of λ values: $(7, 8, 9)$, which is fit for the three levels of MCTF decomposition of the first several GOPs, is used as the starting point for adaptive codec. The fixed λ control codec also uses the same group of lambda values for its motion estimation. The adaptive λ codec allows an adaptation search range $\delta = 2$ for temporal level 1 (i.e., the level with

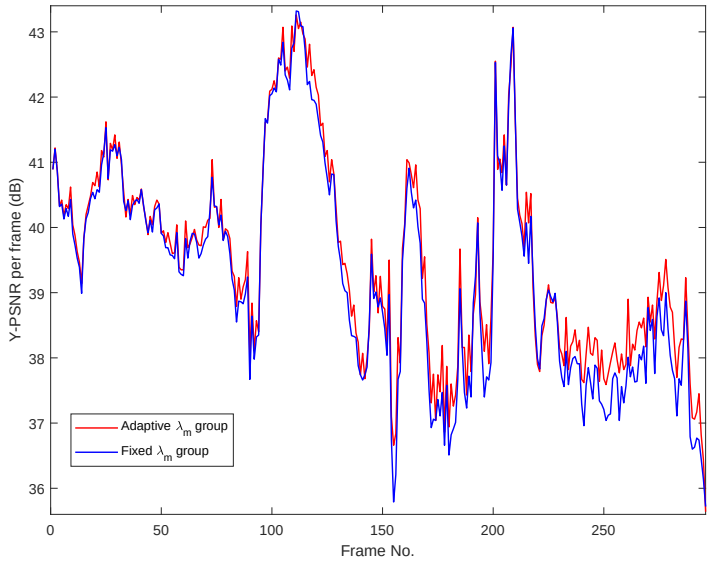

Fig. 33. Y-PSNR results of frame $1-296$ of *Foreman*, obtained by coders with/without motion λ adaptation mechanism

the highest frame rate), δ will be increased by 2 every time when temporal level goes up. We chose 500kbps as a central bitrate for scalability to simulatively decode and check the Y-PSNR gain for each segment. The frame-wise luminance (Y) PSNR gain obtained by adaptive codec is shown in Fig. 33.

We can see the adaptive λ codec can improve the performance in multiple video segments, especially where video feature significantly changes. For frame 151-276 where the camera movement/scene shifting took place, the adaptive codec turned out to achieve an average PSNR improvement of up to 0.25dB by real-timely modifying the Lagrange multipliers for each temporal level.

Fig. 34 further showed how Lagrange multipliers for motion estimation varied since the adaptive motion λ selection started from the third GOP (i.e., frame 17-24) and stopped two GOPs before the end of clip. Unlike other clips such as *Bus* which contains a foreground object with relatively consecutive motion, the first half of *Foreman* mainly contains a human face keeping moving back and forth while the last part is a nearly static scene, thus the feature of motion compensation on different temporal levels may resemble each other, we can see that some GOPs selected quite similar motion λ values for each temporal level. However, the average values of motion λ employed by the second half of the clip appear to be larger than the ones for the first half, which generally encourages the codec to spend more bits in subband coding to obtain better performance.

By changing the central bitrate for simulative decoding as mentioned above, the Y-PSNR performance obtained by code streams extracted with different bitrates can also be shifted. Fig. 35 shows the Y-PSNR curves obtained by fixed motion λ codec and adaptive codecs that chose different simulative decoding bitrates. The Y-PSNR performance obtained by 400kbps group is significantly better within low bitrate range, but will swiftly drop when

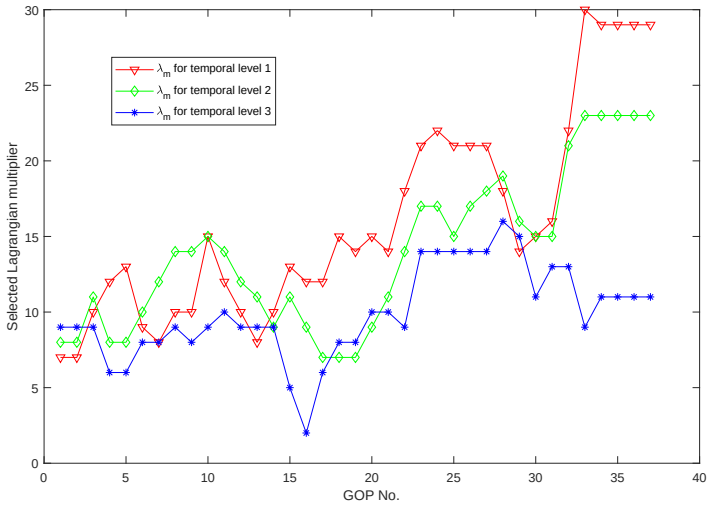

Fig. 34. Best motion Lagrange multipliers for different GOPs/temporal levels

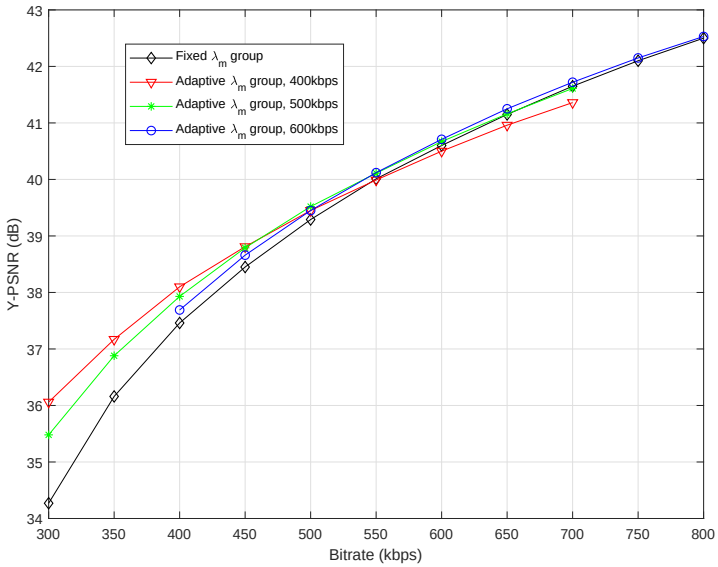

Fig. 35. Overall Y-PSNR curves of *Foreman*, obtained by different codecs.

bitrate is high, while the 600kbps group can achieve stabler performance with higher bitrates by sacrificing the performance under low-bitrate circumstances.

Although such motion λ adaptation based on steepest descent algorithm can effectively improve the overall performance, but the time cost for its implementation is also considerable. Thus we need to optimize the method to make it more practical. We find that remarkable Y-PSNR gain can only be obtained by performing λ adaptation in video seg-

ments that contain significant feature change. Fig. 36 shows the variation of motion λ group as well as the Y-PSNR gain obtained by such change. The x axis shows the GOP 3-34 (frame 17-272) in *Foreman* where adaptation is performed, while y axis shows the Euclidean distance between each λ vector $(\lambda_1, \lambda_2, \lambda_3)$ and the one for previous GOP as well as the obtained Y-PSNR gain measured in 0.01dB. For some specific segments such as GOP 10 and 25, limited λ modification could result in significant Y-PSNR gain and thus is quite efficient, while in most segments the gain is only marginal.

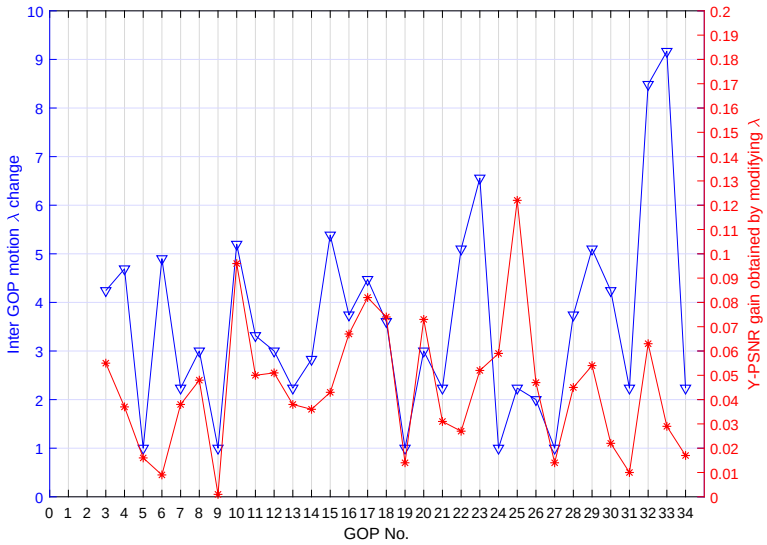

Fig. 36. Change of λ and Y-PSNR gain obtained by adaptation algorithm in processing *Foreman*

Foreman clip is a special instance with abundant, random human expression and camera rotation, which can thus more or less benefit from λ adaptation all the way. A more practical example comes from *Tennis* clip, which contains three distinct segments with relatively stable features: frame 1-24 contains a nearly fixed background and player's waving arm, frame 25-90 mainly contains camera zoom out, a scene change takes place in frame 90 and switches to another scene with fixed background and player's motion. To verify the performance of λ adaptation, we selected lambda group $(8, 15, 24)$, which is fit for the first part of *Tennis*, to be the starting point of adaptation codec as well as the value used for fixed λ control group. The adaptation Y-PSNR gain per frame as well as the overall PSNR versus bitrate curve obtained by frame 1-120 of *Tennis* are as shown in Fig. 37 and 38. Although an overall PSNR gain of up to 0.15dB can be obtained by λ adaptation, but most of them are obtained during processing frame 25-90 (i.e., the distinct segment 2), while the feature of first and third segments are quite similar and the motion estimation is proven to be done pretty well by just using the fixed lambda group.

To know when adaptation mechanism should be turned on and off, certain video features should be monitored and extracted by the encoder. Related discussion as well as subsequent work regarding adaptive motion *lambda* values will be shown in Section 6.

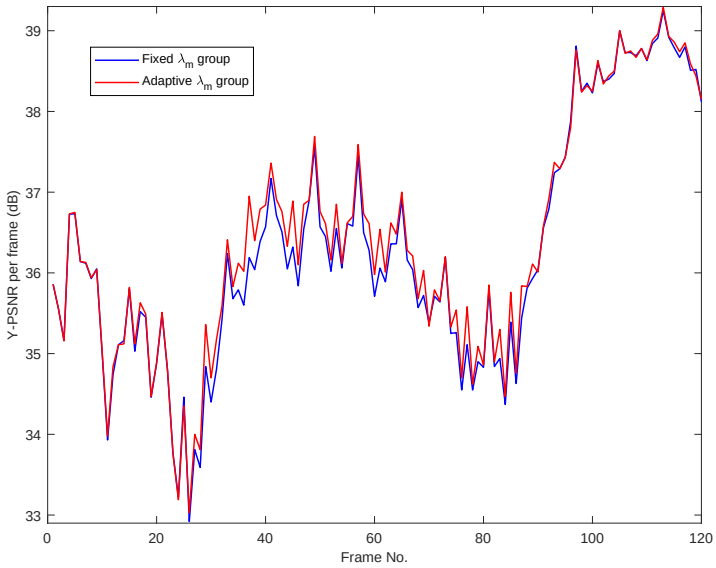

Fig. 37. Y-PSNR results of frame $1-120$ of *Tennis*, obtained by coders with/without motion λ adaptation mechanism

6 Remaining Thesis Work

In Section 3, the framework of Interframe EZBC has been fully developed and may still need more experimental results to show its solidity. On the other hand, the contents of Sections 4 and 5 need to be fully extended to complete their discussion. Therefore, this Section will unfold the work we plan to do regarding these topics. This thesis work should be complete in June 2019.

6.1 Discussion

Compared with AVC/H.264, HEVC/H.265 and ENH-MC-EZBC codecs increased the macro block size for motion compensation from 16×16 to 64×64. Subsequent researchers then further tried larger block sizes when coding HD size videos and obtained obviously better BD-rate performance [45]. In our Interframe EZBC codec, block size should also be increased to fit larger video resolution and enhance the coding performance.

For the multi-component modes in block-based motion compensation, the new MC mode which combines Bergen et al.'s three-frame method for motion-estimation stage with the revised Canny edge detector for completing the whole MCTF processes has been proven to be effective in coding the synthesized video with temporally constant two-component motion pattern. The three new mechanisms: multiple MCTF modes for different motion components, fringe pixel sortation and optional Canny edge selection, have showed their effectiveness in improving the two-component performance.

In Section 3, we have shown the gain achieved by applying MV prediction competition (AMVP) and merging mechanisms to traditional motion compensation modes, which implies two-component motion blocks may also benefit from them. In the test regarding the

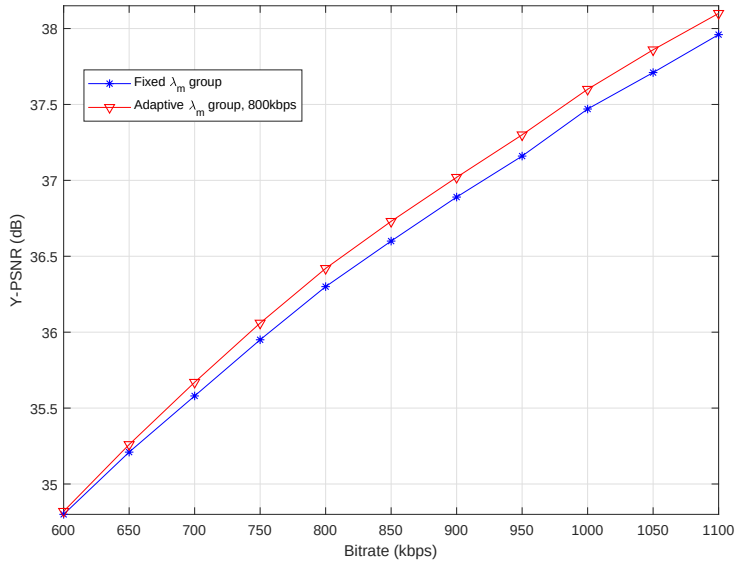

Fig. 38. Overall Y-PSNR curves of *Tennis*, obtained by codecs with/without λ adaptation

synthesized video, we further let the MVs of foreground/background parts to find their respective best predictor in current block's AMVP list and allow these two-component blocks to inherit motion pattern from their neighbors. Fig. 39 shows the preliminary coding result of two-component merging, where two-component blocks are marked by red and green colors (which shows the pixels belonging to foreground/background parts, respectively) and the arrows show the direction of merging. For an ideally moving object, we may describe its motion boundary by only transmitting one block's multi-component MV info and certain merging target index bits. Experimental results showed 12% of motion bytes can be saved by applying AMVP and merging mechanisms to two-component blocks.

Then based on Canny operator, existing results should be extended to form a two-component MC mode with multiple branches/options, where Bergen et al.'s idea based on constant foreground and background motion patterns will be taken as an optional case (or special case) since it resembles the PARALLEL mode as employed in ENH-MC-EZBC codec. Meanwhile, other modes that employ bi-directional or uni-directional reference pictures should be designed to extend two-component idea into common cases. Then a new index for two-component compensation should be added to the mode list of our Interframe EZBC codec as shown in Table 5, whose detailed information such as compensation direction for foreground/background parts, existence of MV residual and etc. will be further indicated by transmitting more control bytes.

Like the traditional block-based motion estimation, multiple possible compensation modes for the foreground/background components may be tried and compared to finally decide on one that optimizes the R-D cost. Last but not least, since affine modes have shown its ability to improve the performance of video coding, we should also attempt to apply affine motion compensation to two-component cases if better R-D performance can be then achieved.

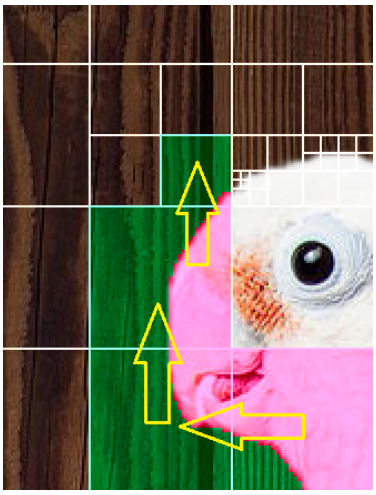

Fig. 39. Mutual merging of two-component motion blocks

Table 5. Motion compensation modes for Interframe EZBC codec with two-component option

Mode No.	Block mode	MCTF direction	Update step	MV residual
1	bi-connected	BI	Yes	2
2	left-connected	LEFT	Yes	1
3	right-connected	RIGHT	Yes	1
4	bi-predicted	BI	No	2
5	left-predicted	LEFT	No	1
6	right-predicted	RIGHT	No	1
7	parallel	BI	Yes	1
8	block-merging	TBD	Yes	0
9	I-block	INTRA	No	0
10	bi-affine	BI	Yes	2
11	left-affine	LEFT	Yes	1
12	right-affine	RIGHT	YES	1
13	two-component	TBD	TBD	TBD

For Section 5 regarding motion λ adaptation, we will continue trying such platform on multiple video clips (especially those with frequent feature change) and compare their PSNR performance with fixed Lagrange multiplier strategies. Besides, we noticed that typical test video clips usually contain only single scene, while real videos include scene change where the video feature as well as rate-distortion balance may also significantly vary, thus we try to concatenate typical video clips into special test sequences with significant scene change points to further simulate practical video scenes and verify the effectiveness of our platform.

As mentioned in Section 5, such mechanism should be turned off when motion estimation feature remains relatively constant to reduce unnecessary time cost, which requires us to extract different parameters regarding interframe/intraframe coding to make decision. Here we just concatenated the first 96 frames (i.e., 12 GOPs) of *Foreman*, *Akiyo* and *Bus* into a new test clip and then process them with the Interframe EZBC codec. The average quadtree-based motion compensation block number of the lowest temporal level in each GOP, which directly reflects the workload and complexity of motion information coding,

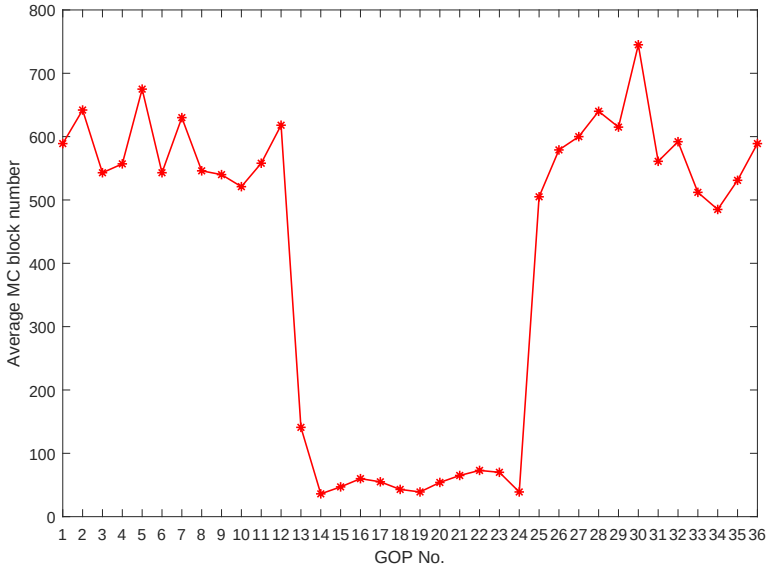

Fig. 40. Average motion-compensation block number in each GOP of the concatenated test clip.

is calculated and shown in Fig. 40 to indicate video feature change. As we can see, the *Foreman* and *Bus* segments showed certain compensation block number change when the feature of human facial expression and vehicle motion changed, while the relatively static scene of *Akiyo* required only limited amount of blocks and remained nearly constant. At the two points where video scenes switched, the abruptly increasing or decreasing compensation block number inform us of absolutely necessary motion λ reselection to enhance the performance. For other segments with significant MCTF feature change, adaptation may also be selectively triggered.

Then we compare the coding performance of multiple adaptive lambda strategies with the traditional fixed lambda scheme, all experimental groups started with motion λ values $(7, 8, 9)$ for three temporal levels, which were fit for the first segment of *Foreman*. The Y-PSNR result of frame 17-288 (where motion λ adaptation is performed) is shown in Fig. 41, where a central bitrate of 600 Kbps was chosen. For the selective adaptation group, motion λ reselection will only be triggered when the average motion block number between two GOPs is above 70. Compared with the full adaptation group (i.e., do adaptation in every GOP) as employed in Section 5, the selective adaptation group achieved nearly the same performance. For the concatenated test clip, adaptation group achieved only 0.05-0.1 dB of Y-PSNR gain in *Foreman* and *Akiyo* segments, since the lambda values we selected for fixed λ group as well as the starting point of adaptation group are fit for the *Foreman* segment while *Akiyo* is nearly static and not sensitive enough to motion λ change. As for the *Bus* segment, two adaptation groups achieved about $0.4dB$ of PSNR gain by wisely reselect Lagrange multipliers.

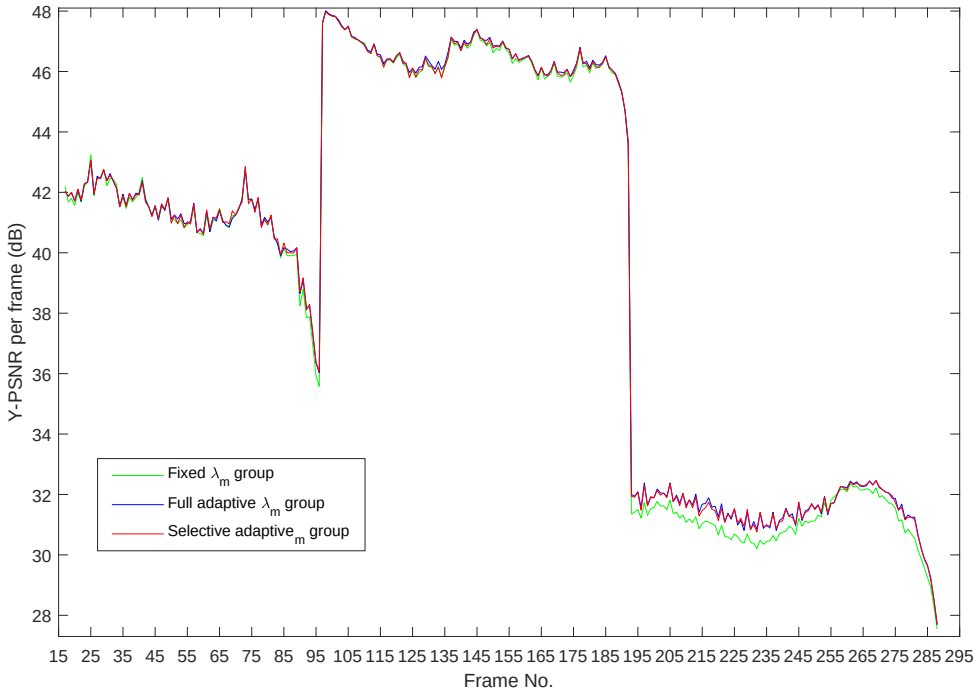

Fig. 41. Y-PSNR results of frame $17-288$ of concatenated test clip, obtained by multiple codecs

6.2 Remaining Thesis Tasks

Task 1: regarding Interframe EZBC codec, the macro block size for motion compensation will be properly increased, then new experimental results of both SD and HD test clips will be further obtained and discussed, the performance of EZBC codec will also be compared with HEVC/H.265.

Task 2: based on the current MCTF framework for two-component motion, motion estimation will not be limited to temporally constant patterns. All kinds of translational motion-compensation ways (forward, backward or bi-directional ones) will be sufficiently developed for two-component mode, foreground/background components will then be allowed to freely select and transmit their own MC modes. New experimental results will then be obtained and compared with different control codecs and discussed.

Task 3: after finishing the work as described in **task 2**, an optional work of trying affine motion patterns in two-component MC blocks should be done to seek for better compensation performance.

Task 4: we will continue extending the adaptive motion λ work to make a trade-off between time cost and coding performance. By taking motion block number as a starting point, various parameters regarding the rate-distortion feature will be extracted to effectively show the trigger points of adaptation. Coding results of original/concatenated test videos by using adaptive and fixed λ mechanisms will be obtained and discussed.

A possible work in our plan: Chen et al. employed fitting function to estimate appropriate motion λ values with respect to each test clip [43] . In our work, video and coding features should then be used to qualitatively or quantitatively predict the Lagrange multipliers on each temporal level when segmental adaptation is triggered, while simulative decoding mechanism will be used to verify and further adjust multiplier, thus significantly improve the efficiency of adaptation mechanism.

References

1. T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, July 2003.
2. M. S. Goldman, "High-Efficiency Video Coding (HEVC): The Next-Generation Compression Technology," in *SMPTE Motion Imaging Journal*, vol. 121, no. 5, pp. 27-33, July 2012.
3. Q. Gong, J. W. Woods, K. Kar and J. Chakareski, "Fine-Grained Scalable Video Caching," *2015 IEEE International Symposium on Multimedia (ISM)*, Miami, FL, 2015, pp. 101-106.
4. Q. Gong, J. W. Woods and K. Kar, "Adaptive MD-FEC over multilink video distribution network," *2016 IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, 2016, pp. 2112-2116.
5. S.-J. Choi and J. W. Woods, "Motion-Compensated 3-D Subband Coding of Video," *IEEE Transactions on Image Processing*, vol. 8, pp. 155-167, Feb. 1999.
6. T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, July 2003.
7. C. M. Huang and M. H. Hung, "Target motion compensation with optical flow clustering during visual tracking," *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control*, Miami, FL, 2014, pp. 96-101.
8. Benjamin Bross, Philipp Helle, Haricharan Lakshman, and Kemal Ugur, "Inter-Picture Prediction in HEVC," In book: *High Efficiency Video Coding (HEVC)*, pp.113-140.
9. G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," in *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74-90, Nov 1998.
10. T. Wiegand, W.-J. Han, B. Bross, J.-R. Ohm, and G. J. Sullivan, "WD1: Working Draft 1 of High-Efficiency Video Coding," document JCTVCC403, Oct. 2010.
11. Guillaume Laroche, Joel Jung, and Beatrice Pesquet-Popescu, "RD Optimized Coding for Motion Vector Predictor Selection," *IEEE Transactions on Circuits And Systems For Video Technology*, VOL. 18, NO. 9, SEPTEMBER 2008.
12. L. Li, H. Li, Z. Lv and H. Yang, "An affine motion compensation framework for high efficiency video coding," *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, Lisbon, 2015, pp. 525-528.
13. Huanbang Chen, Fan Liang and Sixin Lin, "Affine SKIP and MERGE modes for video coding," *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*, Xiamen, 2015, pp. 1-5.
14. Y. Thomas, P. A. Frangoudis and G. C. Polyzos, "QoS-driven multipath routing for on-demand video streaming in a Publish-Subscribe Internet," *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, Turin, 2015, pp. 1-6.
15. Y. Wu and J. W. Woods, "Scalable Motion Vector Coding Based on CABAC for MC-EZBC," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 6, pp. 790-795, June 2007.
16. M. D. Smith and J. Villasenor, "JPEG-2000 Rate Control for Digital Cinema," in *SMPTE Motion Imaging Journal*, vol. 115, no. 10, pp. 394-399, Oct. 2006.

17. H. G. Lalgudi, M. W. Marcellin , A. Bilgin, et al., "Scalable Low Complexity Coder for High Resolution Airborne Video," *International Telemetering Conference Proceedings*. International Foundation for Telemetering, 2007.
18. Peisong Chen and J. W. Woods, "Bidirectional MC-EZBC with lifting implementation," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 10, pp. 1183-1194, Oct. 2004.
19. Shih-Ta Hsiang, "Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling," *Proceedings DCC 2001*. Data Compression Conference, Snowbird, UT, 2001, pp. 83-92.
20. Yongjun Wu, "Fully Scalable Subband/Wavelet Video Coding System", Rensselaer Polytechnic Institute doctoral thesis, August 2005.
21. Y. Wu, K. Hanke, T. Rusert and J. W. Woods, "Enhanced MC-EZBC Scalable Video Coder," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 10, pp. 1432-1436, Oct. 2008.
22. P. Helle et al., "Block Merging for Quadtree-Based Partitioning in HEVC," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1720-1731, Dec. 2012.
23. Peisong Chen; Woods, J.W., "Bidirectional MC-EZBC with lifting implementation," in *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.14, no.10, pp.1183-1194, Oct. 2004.
24. Han Huang; Woods, J.W.; Yao Zhao; Huihui Bai, "Control-Point Representation and Differential Coding Affine-Motion Compensation," in *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.23, no.10, pp.1651-1660, Oct. 2013.
25. R. Mathew and D. S. Taubman, "Hierarchical and Polynomial Motion Modeling with Quad-Tree Leaf Merging," *2006 International Conference on Image Processing*, Atlanta, GA, 2006, pp. 1881-1884.
26. R. Mathew and D. S. Taubman, "Quad-Tree Motion Modeling With Leaf Merging," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 10, pp. 1331-1345, Oct. 2010.
27. Xiaohuan Li, J. R. Jackson, A. K. Katsaggelos and R. M. Mersereau, "An adaptive coding scheme using affine motion model for MPEG P-VOP," *2004 IEEE International Conference on Acoustics*, Speech, and Signal Processing, 2004, pp. iii-717-20 vol.3.
28. Y. Nakaya and H. Harashima, "An iterative motion estimation method using triangular patches for motion compensation," in *Visual Communication and Image Processing*, 1991, vol. 1605, pp. 546C557.
29. H. Huang, J. W. Woods, and Y. Zhao, "Motion compensated prediction using partial mesh generation," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 1677C1680.
30. Robert A. Cohen, "HIERARCHICAL SCALABLE MOTION-COMPENSATED VIDEO CODING," Rensselaer Polytechnic Institute doctoral thesis, January 2007.
31. G. Bjontegaard, "Improvements of the BD-PSNR model," ITU-T Video Coding Experts Group (VCEG), Heinrich-Hertz-Institute, Berlin, Germany, Tech. Rep. VCEG-AI11, Jul. 2008.
32. Tan, T.K., Sullivan, G.J., Wedi, T.: Recommended Simulation Conditions for Coding Efficiency Experiments, ITUT SC16/Q6, document VCEG-AI10 (2008).
33. http://kakadusoftware.com/
34. http://kakadusoftware.com/wp-content/uploads/2014/06/Usage_Examples-v7_9.txt
35. J. R. Bergen, P. J. Burt, R. Hingorani and S. Peleg, "A three-frame algorithm for estimating two-component image motion," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 9, pp. 886-896, Sep 1992.
36. J. Chen, S.R. Lee, K-H. Lee, and W-J. Han, "Object boundary based motion partition for video coding," In *Proc of IEEE International Symposium on Circuits and Systems*, Lisboa, Portugal, Nov. 2007.
37. J. H. Kim, A. Ortega, P. Yin, P. Pandit and C. Gomila, "Motion compensation based on implicit block segmentation," *2008 15th IEEE International Conference on Image Processing*, San Diego, CA, 2008, pp. 2452-2455.

38. O. Divorra Escoda, P. Yin, C. Dai and X. Li, "Geometry-Adaptive Block Partitioning for Video Coding," *2007 IEEE International Conference on Acoustics*, Speech and Signal Processing - ICASSP '07, Honolulu, HI, 2007, pp. I-657-I-660.
39. Canny, J., "A Computational Approach To Edge Detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986.
40. Q. Wang, X. Ji, M. T. Sun, G. J. Sullivan, J. Li and Q. Dai, "Complexity Reduction and Performance Improvement for Geometry Partitioning in Video Coding," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 338-352, Feb. 2013.
41. M. Bläer, C. Heithausen and M. Wien, "Segmentation-based partitioning for motion compensated prediction in video coding," *2016 Picture Coding Symposium (PCS)*, Nuremberg, 2016, pp. 1-5.
42. Q. Huang, H. Wang, S. C. Lim, H. Y. Kim, S. Y. Jeong and C. C. J. Kuo, "Measure and Prediction of HEVC Perceptually Lossy/Lossless Boundary QP Values," *2017 Data Compression Conference (DCC)*, Snowbird, UT, 2017, pp. 42-51.
43. Y. Chen and G. Liu, "Adaptive Lagrange multiplier selection model in rate distortion optimization for 3D wavelet-based scalable video coding," *2014 IEEE International Conference on Image Processing (ICIP)*, Paris, 2014, pp. 3190-3194.
44. Y. Chen, G. Liu, J. Yao, "An improved 3D wavelet-based scalable video coding codec for MC-EZBC," *Multimedia Tools and Applications*, March 2017, Volume 76, Issue 6, pp 7595-632.
45. C. Heithausen and J. H. Vorwerk, "Motion compensation with higher order motion models for HEVC," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, QLD, 2015, pp. 1438-1442.
46. John W. Woods, "Multidimensional Signal, Image, and Video Processing and Coding," Second Edition, Rensselaer Polytechnic Institute, Troy, New York.