Subblock-Based Motion Derivation and Inter Prediction Refinement in the Versatile Video Coding Standard

Haitao Yang, *Member, IEEE*, Huanbang Chen, Jianle Chen, *Senior Member, IEEE*, Semih Esenlik, Sriram Sethuraman, *Senior Member, IEEE*, Xiaoyu Xiu, *Member, IEEE*, Elena Alshina, and Jiancong Luo

(Invited Paper)

***Abstract*— Efficient representation and coding of fine-granular motion information is one of the key research areas for exploiting inter-frame correlation in video coding. Representative techniques towards this direction are affine motion compensation (AMC), decoder-side motion vector refinement (DMVR), and subblock-based temporal motion vector prediction (SbTMVP). Fine-granular motion information is derived at subblock level for all the three coding tools. In addition, the obtained inter prediction can be further refined by two optical flow-based coding tools, the bi-directional optical flow (BDOF) for bi-directional inter prediction and the prediction refinement with optical flow (PROF) exclusively used in combination with AMC. The aforementioned five coding tools have been extensively studied and finally adopted in the Versatile Video Coding (VVC) standard. This paper presents technical details of each tool and highlights the design elements with the consideration of typical hardware implementations. Following the common test conditions defined by Joint Video Experts Team (JVET) for the development of VVC, 5.7 % bitrate reduction on average is achieved by the five tools. For test sequences characterized by large and complex motion, up to 13.4 % bitrate reduction is observed. Additionally, visual quality improvement is demonstrated and analyzed.**

***Index Terms*— Versatile video coding (VVC), inter prediction, affine motion compensation (AMC), decoder-side motion vector refinement (DMVR), subblock-based temporal motion vector prediction (SbTMVP), bi-directional optical flow (BDOF), prediction refinement with optical flow (PROF).**

I. Introduction

VIDEO coding standards play an increasingly important role in diversified video applications and services ranging from the conventional television broadcasting, internet protocol television (IPTV), digital cinema and surveillance, to the internet-based ones including over-the-top (OTT) video, social media with short video sharing, and video conferencing with screen sharing. The Versatile Video Coding (VVC) standard [1] was developed by the Joint Video Experts Team (JVET) to face the technical challenge of higher compression ratio for the exponentially increasing traffic of video content and to meet the requirements from emerging markets including the products and services with featured content such as high dynamic range (HDR), augmented reality (AR) and virtual reality (VR), etc. With the classical block-based hybrid video coding architecture, VVC boosts the rate-distortion (RD) performance of each module significantly and achieves an overall bit rate reduction of more than 40 % with the same reproduction quality measured by peak signal-to-noise ratio (PSNR) for ultra-high definition (UHD) content.

Exploiting the inter-frame correlation, inter predictive coding is one of the main sources of coding gain and evolves in several aspects through the generations of video coding standards. For inter coding in the High Efficiency Video Coding (HEVC) standard [2], the predecessor of VVC standard, the prediction of samples in a prediction unit (PU) are derived from one or two reference pictures. Two lists of pictures in the decoded picture buffer are constructed, denoted as list 0 and list 1, and are used for the management of the reference pictures. For a PU in P slices, only uni-prediction can be applied, where the motion-compensated prediction is derived from one reference picture in list 0. For a PU in B slices, uni-prediction may be applied using a reference picture in either list 0 or list 1. Furthermore, bi-prediction can be optionally applied to derive the motion-compensated prediction from two reference pictures, one in list 0 and the other in list 1. Motion information of a PU is used to perform motion compensation (MC) to derive the prediction for the samples of the PU. Motion information typically consists of an indication of prediction direction, one or two motion vectors, and one or two reference indices associated with each motion vector (MV). The prediction direction indicates whether uni-prediction or bi-prediction is used. And in case of uni-prediction, whether list 0 or list 1 is used. Only one MV is required for uni-prediction, along with the associated reference index indicating the exact reference picture in the list. Two MVs and associated reference indices are required for bi-prediction. It should be noted that the indication of the prediction direction is not

Manuscript received August 22, 2020; revised March 25, 2021 and June 17, 2021; accepted July 17, 2021. Date of publication July 27, 2021; date of current version October 4, 2021. This article was recommended by Associate Editor J.-R. Ohm. *(Corresponding author: Haitao Yang.)*

Haitao Yang and Huanbang Chen are with Huawei Technologies Company Ltd., Shenzhen 518129, China (e-mail: haitao.yang@huawei.com; chenhuanbang@huawei.com).

Jianle Chen is with Qualcomm Technologies Inc., San Diego, CA 92121 USA (e-mail: cjianle@qti.qualcomm.com).

Semih Esenlik and Elena Alshina are with Huawei Technologies Duesseldorf GmbH, 80992 Munich, Germany (e-mail: semih.esenlik@huawei.com; elena.alshina@huawei.com).

Sriram Sethuraman is with Prime Video Playback Team, Amazon, Bengaluru 560052, India (e-mail: sriram.sethuraman@protonmail.com).

Xiaoyu Xiu is with Kwai Inc., San Diego, CA 92122 USA (e-mail: xiaoyuxiu@kwai.com).

Jiancong Luo is with Apple Inc., Cupertino, CA 95014 USA (e-mail: jluo8@apple.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2021.3100744.

Digital Object Identifier 10.1109/TCSVT.2021.3100744

required for PUs in P slices where only uni-prediction is possible. Similarly, the reference index is not required if there is only one reference picture in the list.

The block merging technique was incorporated in HEVC as the merge mode for coding the motion information in a more efficient way [3]. In the merge mode, a merge list with several motion information candidates is constructed for a PU. Candidates are primarily derived from spatial and temporal neighboring blocks. An index to the selected candidate instead of the candidate motion information itself is transmitted so the transmitting overhead can be significantly reduced. The MV can also be explicitly coded in case the merge mode cannot provide an accurate motion information. Similar to the merge mode, in HEVC MV coding [4], the encoder firstly derives two MV candidates and then transmits the index of the selected one. With the competition of two MV predictors, the cost of transmitting the MV difference can be greatly reduced. The MV predictor is scaled according to the temporal distance between the current picture and the reference picture of the current PU, if necessary, to guarantee that it originates from the current picture and points to the reference picture.

VVC inherits the same basic concepts and mechanisms of the block merging and the MV coding techniques from HEVC, and improves them in many aspects to achieve significant compression efficiency enhancement [5]. Variable block size for MC is another key element of inter prediction to match the boundary of moving objects and therefore can reduce the prediction residual around the boundary. The structure of quad-tree with nested multi-type tree using binary and ternary splits is introduced in VVC for partitioning a coding tree unit (CTU) into multiple coding units (CUs) [6], where a CU comprises one luma coding block (CB) and ordinarily two chroma CBs with associated syntax elements. It is worth noting that there is no PU in VVC, since the syntax for prediction operations is specified at CU level. For more efficiently coding video content with a large spatial resolution such as 4K and 8K, the size of CUs can vary from 128×128 down to 4×8 or 8×4 for inter coding in VVC. One reason of not having a smaller CU size, e.g., 4×4 CUs for inter coding, is the dissatisfactory compression performance due to the expensive transmitting overhead of fine-granular MVs. Furthermore, relatively large block sizes will be decided by a typical rate-distortion optimized encoder for the trade-off of the transmitting overhead and the distortion, and therefore stronger blocking artifacts along block boundaries can be observed. Various inter coding technologies have been proposed during the VVC standardization process to improve the overall rate-distortion performance and the visual quality especially at low bitrate coding.

Leveraging high-order deformation models for representing fine-granular motion between the current picture and its reference picture is expected to achieve a high-quality prediction with few side information of model parameters. As a representative high-order model, the affine model is extensively studied in combination with existing video coding standards [7]–[10]. In [11], [12] the classic 6-parameter affine model is simplified to a 4-parameter one for less overhead in signaling model parameters. It is shown that the combination of translation, zooming, and rotation, which can be represented by the 4-parameter model, is already a good approximation of complex inter-frame motion. The affine motion compensation (AMC) solution based on the 4-parameter model demonstrated significant coding gain [11], and was adopted in the first version of the Joint Exploration Model (JEM) [13] in 2015. It was later refined in many aspects including alternative 6-parameter model [14], model inheritance, model construction [15], control point motion vector coding [16], complexity reduction [17]–[22], and evolved to the design in VVC [23]. It is worth noting that both luma and chroma CBs of a CU are split into 4×4 subblocks for motion vector derivation and motion compensation [19] in VVC AMC. In contrast to the sample-wise processing in traditional affine schemes, the introduction of subblocks not only makes a good trade-off of coding efficiency and computational complexity, but also reduces the implementation cost by reusing existing hardware or software modules for block-based motion compensation.

Another way of reducing the overhead of transmitting motion parameters is to derive motion parameters at the decoder side [24]–[28]. Based on the assumption of constant motion, bilateral matching (BM) [24], [25] estimates the displacement between two reference pictures and derives the motion vectors for the current picture that is in between the two reference pictures. Template matching (TM) obtains an L-shaped template around the current block from the reconstructed picture region and performs template matching in the reference picture to derive the motion vector for the current block [26]–[28]. The frame rate up-conversion (FRUC) [29] in JEM incorporates both BM and TM as alternative merge modes, where the luma block of a CU is partitioned into subblocks and the motion vectors are derived for each one. The concepts of BM and TM are combined in [30], where the template is constructed by averaging two temporal neighboring blocks and is used to perform a search in the two reference pictures. This decoder-side motion refinement (DMVR) extends the existing merge mode [3] with a low-complexity motion refinement and does not need to transmit additional side information. With delicate design for harnessing the computational complexity in several aspects, the DMVR scheme in [31] was adopted to VVC. The subblock partition, among various modifications on top of JEM DMVR [13], is considered to be an important feature to tackle practical difficulties in typical hardware implementations.

The subblock concept can also be applied to extend the CU level temporal motion vector prediction (TMVP) in the merge mode [3]. The Subblock-based TMVP (SbTVMP) was initially proposed in the exploration stage for the new video coding standard after HEVC [32] and was adopted in JEM. It allows inheriting the motion information at subblock level from the collocated reference picture. With this design, each subblock of a large size CU can have its own motion information without explicitly transmitting the block partition structure or motion information, enabling diverged motion compensation at no cost.

During the development of HEVC standard [2], one technique called bi-directional optical flow (BDOF) was proposed [33]–[35] to compensate the sample-wise fine motion

that is missed by the block-based motion compensation. Specifically, the motion refinements are derived implicitly from the samples of two prediction blocks based on the optical flow differential equation. This method shows substantial coding performance improvement and was adopted into JEM. It was further simplified in several aspects including early termination strategy [36], internal bit depth decrease [37], low-complexity gradient calculation and motion refinement [38], and was adopted into VVC at the 12^{th} JVET meeting.

Optical flow can also be applied to the subblock-based AMC to achieve the effect of sample-wise MC [39]. As an input to the optical flow equation, the motion refinement for each sample is obtained by subtracting the MV at the subblock center from the MV at the sample location where both MVs are derived from the same affine model. This design is referred to as prediction refinement with optical flow (PROF) in VVC [40].

This paper aims to provide a brief yet comprehensive introduction of the abovementioned five subblock-based inter coding tools in VVC standard, and is organized as follows. Section II presents the details of AMC, DMVR and SbTMVP, especially the mechanism of subblock-based motion derivation. Section III describes BDOF and PROF that target at quality enhancement of the prediction in a post-refinement stage. Section IV highlights design elements to handle potential difficulties in practical implementations. Section V presents simulation results and provides an analysis thereof. Section VI concludes the paper.

II. Subblock-Based Motion Derivation

One commonality among subblock-based motion derivation mechanisms in AMC, DMVR, and SbTMVP is to derive luma MV with $1/16$ fractional-sample accuracy for subblock MC, instead of explicit coding subblock MVs. This feature enables subblock level MC at the cost of CU level signaling of motion information, which can theoretically reduce the number of bits for motion information coding for the same quality of inter prediction.

A. Affine Motion Compensation

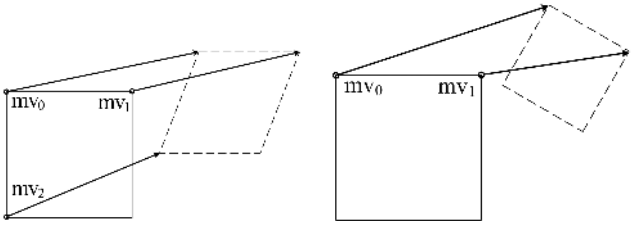
There are two variants of AMC in VVC, the affine inter mode and the affine merge mode. The difference is that the affine motion model is explicitly transmitted for the former one but derived at the decoder side for the latter one.

1) Control Point Representation for Affine Motion Model: Affine motion model of two types, a 4-parameter one and a 6-parameter one, are introduced for describing complex motion typically characterized by zooming and rotation in addition to translation. The classic 6-paramater affine model is

$$\begin{aligned} mv_x &= a * i + c * j + e \\ mv_y &= b * i + d * j + f, \end{aligned} \tag{1}$$

where $mv = (mv_x, mv_y)$ is the MV at coordinate (i, j); a, b, c, d, e, and f are the model parameters.

Representing the affine model by control point motion vectors (CPMVs) instead of the conventional model parameters was adopted in VVC AMC to take the advantage of the existing prediction and the block merging techniques for MV coding. As shown in Fig. 1(a), the CPMVs representing a 6-parameter affine model for a CU are defined as the MVs at the top-left, top-right and bottom-left corner luma sample positions of the CU, namely the control points of the CU. And the MV at coordinate (i, j) of the luma CB is described as



(a) 6-parameter affine (b) 4-parameter affine

Fig. 1. Control point based affine motion model.

$$\begin{aligned} mv_x(i, j) &= \frac{mv_{1,x} - mv_{0,x}}{W} i + \frac{mv_{2,x} - mv_{0,x}}{H} j + mv_{0,x} \\ mv_y(i, j) &= \frac{mv_{1,y} - mv_{0,y}}{W} i + \frac{mv_{2,y} - mv_{0,y}}{H} j + mv_{0,y}, \end{aligned} \tag{2}$$

where $mv_k = (mv_{k,x}, mv_{k,y})$, $0 \leqslant k \leqslant 2$ are the MVs of top-left, top-right, and bottom-left corner control points. W and H are the width and height of the CU.

Applying the constraints of $a = d$ and $b = -c$ in (1), the 6-paramater affine model is simplified to a 4-paramater one,

$$\begin{aligned} mv_x &= a * i - b * j + e \\ mv_y &= b * i + a * j + f. \end{aligned} \tag{3}$$

Similarly, as shown in Fig. 1(b), the 4-parameter model is represented by two CPMVs at the top-left and top-right control points of the CU.

The luma CB of a CU is split into 4×4 subblocks. The MV at the central sample position of a subblock, with coordinates as $(2, 2)$ specifically, is calculated according to the affine motion model and set as the subblock MV, as illustrated in Fig. 2. The calculated subblock MV is rounded to $1/16$ fractional-sample accuracy as the output. With the derived subblock MV, a set of 6-tap interpolation filters is applied to generate the prediction of each subblock [22]. The subblock size of chroma components is set to be 4×4, and the MV of a 4×4 chroma subblock is calculated as the average of the MVs of the top-left and bottom-right luma subblocks in the collocated 8×8 luma region for video contents in $4 : 2 : 0$ color format [19].

2) Affine Inter Mode: Affine inter mode is restricted to CUs with both width and height larger than or equal to 16 luma samples, because the cost of transmitting CPMVs for smaller size CUs is too high to achieve a reasonable trade-off in rate-distortion performance. The usage of the mode is controlled by an indication flag. Another flag is signaled to indicate the motion model type, i.e., whether the 4-parameter model or the 6-parameter model is used. Depending on the motion model type, two or three motion vector differences (MVDs)

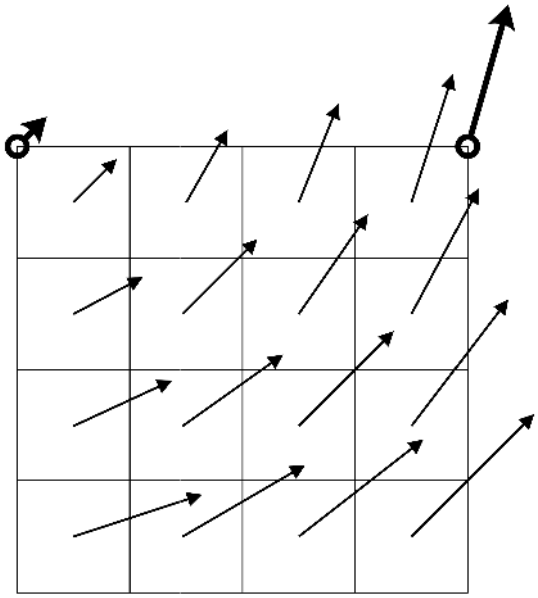


Fig. 2. Subblock motion field for affine.

between the CPMVs and the predictors of them are signaled. Without loss of generality, the 6-parameter model is used for explanation hereinafter in this section. The advanced motion vector prediction mechanism for MV coding in VVC [5] is adapted to the affine inter prediction where an affine motion vector predictor (MVP) comprises three MVPs at the three control points of the CU. For the derivation of the affine MVP in the prediction direction of the current CU, an affine MVP list with two candidates is constructed and an index is signaled to indicate the one selected at the encoder side. Note that this affine MVP derivation process is applied twice, separately for list 0 prediction and list 1 prediction, when bi-prediction is applied to the current CU.

The affine MVP list is constructed by firstly inserting the left neighbor candidate and the above neighbor candidate if available. The model inheritance mechanism is introduced to derive the affine model of the current CU from a spatial neighboring CU coded with affine mode. Specifically, the affine model of a neighbor CU coded in affine mode, represented by three CPMVs of (mvN_0, mvN_1, mvN_2), is applied to derive the MVPs at the three control points of the current CU, as shown in Fig. 3. The affine MVP candidate derived in this way is called an inherited affine MVP candidate. The neighboring CUs in position A0 and A1 are checked in order and the model inheritance mechanism is applied to the first available affine-coded CU to derive the left neighbor candidate. Note that the left neighbor candidate may not exist if the CUs at A0 and A1 are not affine-coded or the reference picture of the neighbor affine-coded CU is not the same as that of the current CU. The same process of checking availability and candidate derivation is applied to obtain the above neighbor candidate from CUs at B0, B1, B2 positions.

Secondly, the model construction mechanism is employed to derive one constructed affine merge candidate and put it in the affine MVP list if the list is not yet full. As illustrated in Fig. 4, the spatial neighbors around each control point are checked one by one, to identify if there exists an MV that

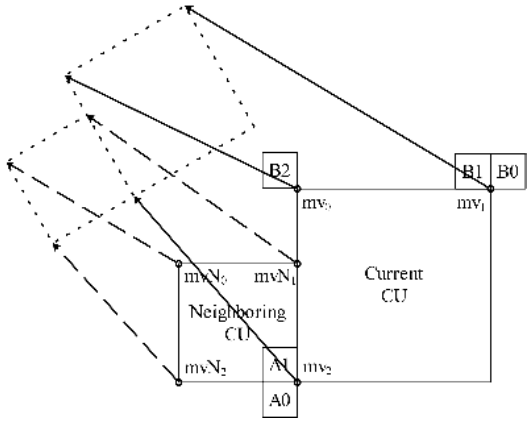


Fig. 3. Affine motion model inheritance.

points to the same reference picture as the current CU, and the first available MV is assigned as the CPMV predictor of the control point. The checking order is B2→B3→A2 for CP1, B1→B0 for CP2 and A1→A0 for CP3. The CPMV predictors at CP1, CP2 and CP3 are denoted as mvp_0, mvp_1, and mvp_2, respectively. A constructed affine candidate is obtained if all the three CPMV predictors are available.

If the affine MVP list is still not full, stuffing affine candidates are inserted to the list. A stuffing affine candidate is derived by setting all three CPMV predictors to one of the MV in a pre-defined set of stuffing candidates. The pre-defined set of stuffing candidates comprises mvp_2, mvp_1, mvp_0, TMVP, and zero MV. Each candidate in the set can only be used once.

The CPMVs (mv_0, mv_1, mv_2) of the 6-parameter model are derived at the decoder side as

$$\begin{aligned} mv_0 &= mvp_0 + mvd_0 \\ mv_1 &= mvp_1 + mvd_1 + mvd_0 \\ mv_2 &= mvp_2 + mvd_2 + mvd_0, \end{aligned} \tag{4}$$

where (mvp_0, mvp_1, mvp_2) is the affine MVP candidate selected at the encoder side, mvd_0, mvd_1, and mvd_2 are the received MVDs for the three MVPs accordingly. As illustrated in (4), both mvd_1 and mvd_2 are predicted by mvd_0. This MVD prediction mechanism is designed specifically for coding affine MVDs and can exploit the similarities in the MVDs at affine control points.

3) Subblock Merge Mode: Similar to the regular merge mode in VVC where a list of motion candidates is derived for inter prediction at CU level [5], the subblock merge mode in VVC constructs a separate merge list with only subblock-based motion candidates. The candidate selected by the encoder is indicated by a merge index which is transmitted to the decoder. Subblock merge mode is applied to CUs with both width and height larger than or equal to 8 luma samples [41]. The SbTMVP candidate (introduced in Section II-B) is put in the first place of the list, followed by inherited affine merge candidates and constructed affine merge candidates. Maximum 5 candidates can be put in the subblock merge list. Since most of the candidates in the list are affine merge candidates, the subblock merge mode may also be referred to as the affine merge mode.

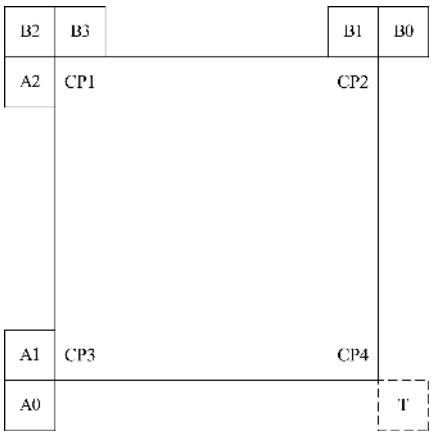


Fig. 4. Positions for the derivation of constructed affine MVP candidates and constructed affine merge candidates.

Similar to the derivation of the inherited affine MVP candidate, the availability of the left and the top neighbor candidates are checked to derive the inherited affine merge candidates. The only difference is that the spatial neighboring CU in affine mode is not required to have the same reference picture as the current CU. The reference index of the current CU is set to be the same as that of the neighboring affine-coded CU, in the derivation of the inherited affine merge candidates. In addition, the motion model type and the prediction direction of the neighboring affine-coded CU are reused as well.

For the derivation of the constructed affine merge candidates, the motion information of all four control points is firstly derived from the spatial and temporal neighbors, as shown in Fig. 4. Using the same checking order as in the derivation of the constructed affine MVP candidate, the motion information at CP1, CP2 and CP3 is obtained. The temporal motion information in the collocated position T is obtained and set to be the motion information at CP4. The motion information at a control point may not exist if no valid motion information is found, e.g., in case the neighboring CU is coded in intra mode. A set of control point combinations is constructed as {CP1, CP2, CP3}, {CP1, CP2, CP4}, {CP1, CP3, CP4}, {CP2, CP3, CP4}, {CP1, CP2}, {CP1, CP3}, and is checked one after another. The availability of all CPMVs of a control point combination in bi-prediction, list 0 prediction, and list 1 prediction are checked in order, and a valid affine merge candidate is constructed if all CPMVs in a specified prediction direction exist and share the same reference index. Note that only the first valid candidate constructed from control point combination is selected. It is obvious that a 6-parameter affine motion model will be constructed from the combinations with three control points, and a 4-parameter model will be derived from the combinations with two control points.

If the subblock merge list is still not full after adding all subblock merge candidates abovementioned, one or more stuffing candidates of the 4-parameter affine model with zero CPMVs are inserted into the list.

B. Subblock-Based Temporal Motion Vector Prediction

SbTMVP obtains motion information for each subblock of a CU in three steps: a) derive the displacement vector (DV) for the current CU, b) check availability of the SbTMVP candidate and derive the central motion, and c) derive the subblock motion information from the corresponding subblock identified by the DV. The derived subblock level motion information is used for the MC of both luma and chroma CBs of the CU. The collocated picture is the reference picture that is used as the source picture for temporal motion information derivation. Unlike TMVP candidate derivation in HEVC merge mode [3], which always derives the temporal motion vectors from the collocated block in the collocated picture, SbTMVP applies a DV to find the correspondence of the positions or the partitioned blocks in the current picture and those in the collocated picture. As shown in Fig. 5, the MV of the left neighboring CU of the current CU is selected to be the DV if the left neighboring CU uses the collocated picture as its reference picture. In case the left neighboring CU is not coded in inter prediction mode or the MV does not point to the collocated picture, the DV is set to $(0, 0)$. The DV is then applied to the central position of the current CU to locate the displaced central position in the collocated picture, as illustrated in Fig. 5. If the block containing the displaced central position is not inter-coded, the SbTMVP candidate is considered not available. Otherwise, the motion information of the central position of the current CU, named as central motion, is derived from the motion information of the block containing the displaced central position in the collocated picture, marked as MV3 in Fig. 5. The central motion is derived in a similar way as the temporal motion candidate derivation where the temporal motion scaling is applied to align the reference pictures of the temporal motion vectors to those of the current CU [4]. Up to two motion vectors, one per list, can be derived. When the SbTMVP candidate is available, the DV is applied to find the corresponding subblock in the collocated picture for each subblock of the current CU. And the motion information of the corresponding subblock is used to derive the motion information of the subblock in the current CU in the same way as deriving the central motion. In case the corresponding subblock is not inter-coded, the motion information of the current subblock is set to be the central motion.

SbTMVP is applicable to CUs with both width and height larger than or equal to 8 luma samples. The subblock size for SbTMVP is set to 8×8 in VVC [42] in order to restrict the memory bandwidth consumption not exceeding the worst-case situation in 8×8 bi-prediction. To avoid additional memory access burden of loading motion information from random locations in the collocated picture, the location is restricted to be within the collocated area of the current CTU plus one column of 4×4 blocks at the right boundary. The location of the corresponding subblock is clipped to be within the constrained area if it goes outside after applying the DV.

C. Decoder-Side Motion Vector Refinement

This section presents the design of DMVR in VVC. Readers can refer to [43]–[45] for the details of the technical discussion and the alternative versions of DMVR considered along its evolution.

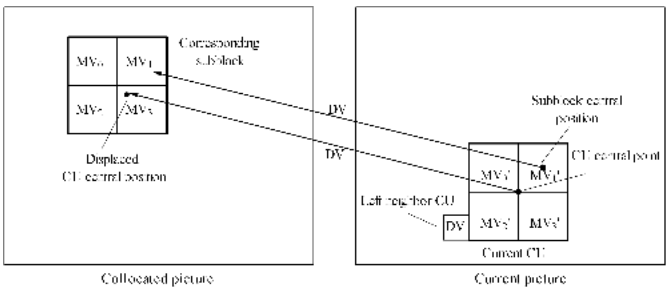


Fig. 5. Derivation of the SbTMVP candidate.

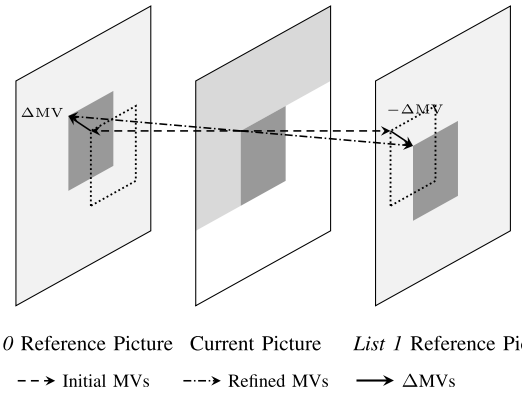


Fig. 6. Bilateral matching-based DMVR and ΔMV mirroring property (adapted from [45], © 2020 IEEE).

1) General Description of DMVR Process: DMVR is applied to CUs in the regular merge mode in VVC [5]. The pair of MVs obtained from the regular merge mode is the input to the DMVR process. As shown in Fig. 6, DMVR applies bilateral matching to refine the accuracy of the input MV pair $\{mv0_{0,0}, mv1_{0,0}\}$ and uses the refined MV pair for the motion-compensated prediction of both luma and chroma CBs of a CU. The input and the output of the DMVR process, namely the initial and refined MV pairs, obey the following equation:

$$\begin{aligned} mv0_{\text{refined}} &= mv0_{0,0} + \Delta mv, \\ mv1_{\text{refined}} &= mv1_{0,0} - \Delta mv. \end{aligned} \tag{5}$$

The motion vector difference Δmv is added to $mv0_{0,0}$ and subtracted from $mv1_{0,0}$ to obtain the refined MV pair $\{mv0_{refined}, mv1_{refined}\}$, which is known as the MVD mirroring property of DMVR. MVD mirroring property is applied to reduce the number of candidate MV pairs, and is frequently used in VVC for MVD coding, for instance, in the merge mode with MVD and the symmetric MVD coding mode [5]. To ensure the equal distance MVD mirroring property, DMVR is allowed only if the initial MV pair point to two different reference pictures that have equal distance in picture order count (POC) to the current picture, wherein POC is an integer value used to uniquely identify each picture in a coded sequence. In addition to the equal POC distance restriction as described above, DMVR is restricted to CUs with width and height larger than or equal to 8 luma samples and the number of luma samples greater than or equal to 128, to avoid DMVR process on smaller CUs for complexity reduction. Other restrictions on the usage of DMVR can be found in [23].

A luma CB is divided into 16×16 subblocks for the MV refinement process if both the width and height of the CB are greater than or equal to 16 luma samples. If the width or the height of a luma CB is equal to 8 samples, the subblock size is set to be 8×16 or 16×8, respectively. The Δmv is obtained independently for each subblock in two steps, an integer-sample search step followed by a fractional-sample search step. And finally the subblock MC is applied using $\{mv0_{\text{refined}}, mv1_{\text{refined}}\}$.

2) Integer-Sample Search: A search space consisting of 25 candidate MV pairs is constructed as follows:

$$\begin{aligned} mv0_{i,j} &= mv0_{0,0} + 16\langle i, j\rangle, \\ mv1_{i,j} &= mv1_{0,0} - 16\langle i, j\rangle. \end{aligned} \tag{6}$$

where $\langle i, j\rangle$ represents the coordinate of a search point around the initial MV pair, and i and j are integer numbers between -2 and 2 inclusive. Since the internal MV precision is in 1/16 fractional-sample in VVC, the difference vector $\langle i, j\rangle$ is multiplied by 16. The difference between the candidate MVs are multiple of an integer-sample interval, hence the first step of DMVR is called the integer-sample search.

The candidate MV pairs are used to obtain pairs of motion-compensated luma prediction blocks (denoted $P0_{i,j}$ and $P1_{i,j}$). A bilinear interpolation filter, which has a lower memory bandwidth requirement and computational complexity than the 8-tap DCT-based interpolation filter (DCT-IF) [46], is applied at this stage. Test results in [47] and [48] had shown a negligible RD performance degradation when using bilinear interpolation instead of DCT-IF. It should be noted that the bilinear interpolation can be performed at CU level, as all subblocks in a CU have the same search space.

Afterwards, row subsampled sum of absolute differences (SAD) cost is calculated for prediction blocks of each candidate pair according to

$$SAD(i, j) = K \sum_{n=0}^{\frac{H}{2}} \sum_{m=0}^{W} diff_{m,n}, \tag{7}$$

$$\begin{aligned} diff_{m,n} = \text{abs}(&P0_{i,j}[m + i, 2n + j] \\ &- P1_{i,j}[m - i, 2n - j]), \end{aligned} \tag{8}$$

where

$$K = \begin{cases} 3/4 & i = 0, j = 0 \\ 1 & \text{otherwise.} \end{cases} \tag{9}$$

Here, W and H are the width and height of the current subblock. The row subsampling reduces the operations for SAD computation by a factor of 2, while still being friendly to Single Instruction Multiple Data (SIMD) operations. The SAD corresponding to the initial MV pair is scaled with a factor of 3/4 to favor the center point in the search window, in order to stabilize the refinement process. The search coordinates resulting in the minimum SAD cost, denoted as $\langle i_{min}, j_{min}\rangle$, is selected as the output of the integer-sample search step.

The SAD cost with the initial MV pair, indicating the center position in the search space, is first computed. Only if this cost

is greater than or equal to a threshold value that is equal to the number of samples in the subblock, the remaining SAD costs are evaluated. This early termination results in complexity reduction and power savings in both hardware and software implementations with negligible impact on coding gains.

3) Fractional-Sample Search: The candidate MV pair selected in the integer-sample search step is further refined by leveraging the SAD costs already calculated. A quadratic error surface function with the following form, which is depicted in Fig. 7, is used to model the SAD costs at fractional-sample search coordinates:

$$SAD(x, y) = \alpha(x - x_{min})^2 + \beta(y - y_{min})^2 + \gamma. \quad (10)$$

Equation (10) is fitted to 5 of the 25 SAD costs calculated in the first step in order to determine the 5 unknowns parameters, i.e., α, β, γ, x_{min}, and y_{min}. The 5 SAD costs are selected as the costs corresponding to the coordinates $\langle i_{min}, j_{min}\rangle$, $\langle i_{min}-1, j_{min}\rangle$, $\langle i_{min}, j_{min}-1\rangle$, $\langle i_{min}+1, j_{min}\rangle$ and $\langle i_{min}, j_{min}+1\rangle$. The unknown parameters α, β, x_{min}, and y_{min} are therefore determined as

$$\alpha = SAD_x - 2SAD(i_{min}, j_{min}), \quad (11)$$

$$\beta = SAD_y - 2SAD(i_{min}, j_{min}), \quad (12)$$

$$x_{min} = \begin{cases} 16i_{min} + \lfloor 8\frac{SAD_\alpha}{\alpha}\rfloor & \alpha \neq 0 \\ 16i_{min} & \alpha = 0, \end{cases} \quad (13)$$

and

$$y_{min} = \begin{cases} 16j_{min} + \lfloor 8\frac{SAD_\beta}{\beta}\rfloor & \beta \neq 0 \\ 16j_{min} & \beta = 0 \end{cases} \quad (14)$$

with

$$SAD_x = SAD(i_{min} - 1, j_{min}) + SAD(i_{min} + 1, j_{min}), \quad (15)$$

$$SAD_y = SAD(i_{min}, j_{min} - 1) + SAD(i_{min}, j_{min} + 1), \quad (16)$$

$$SAD_\alpha = SAD(i_{min} - 1, j_{min}) - SAD(i_{min} + 1, j_{min}), \quad (17)$$

and

$$SAD_\beta = SAD(i_{min}, j_{min} - 1) - SAD(i_{min}, j_{min} + 1). \quad (18)$$

The motion vector difference in fractional-sample accuracy is determined by the coordinates $\langle x, y\rangle$ that minimizes the SAD cost $SAD(x, y)$, which is equal to $\langle x_{min}, y_{min}\rangle$. The Δmv in (5) is set equal to $\langle x_{min}, y_{min}\rangle$, which is then used to obtain the refined motion vector pairs, i.e., $mv0_{refined}$ and $mv1_{refined}$. It is worth noting that this fractional-sample search step will be skipped if any of the absolute values of i_{min} and j_{min} is equal to 2, indicating that $\langle i_{min}, j_{min}\rangle$ is at the boundary in the search space of the integer-sample search step. Because in this case not all of the 5 SAD costs required for the fractional-sample search would be available.

This parametric error surface function based fractional-sample search eliminates the need for any sample processing in this stage. Against an alternative method that performs one iteration of half-sample refinement, this method offers a decoder complexity reduction (measured by running time) of 7 % without compression efficiency penalty [49].

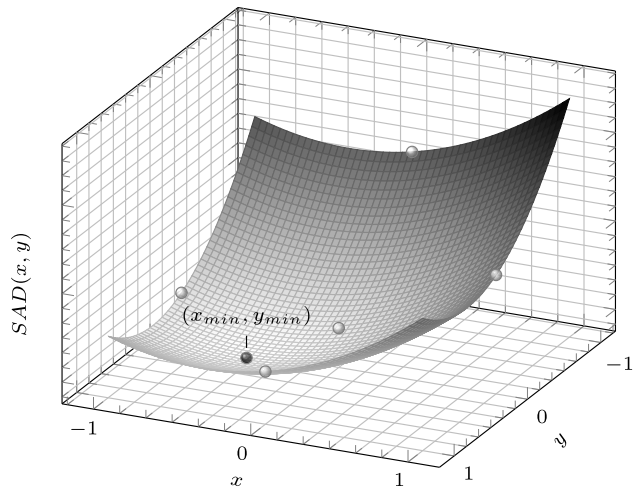


Fig. 7. Quadratic error function based SAD cost surface model (adapted from [45], © 2020 IEEE).

III. Prediction Refinement With Optical Flow

Compared to traditional block MC, optical flow is expected to achieve the effect of sample-wise inter prediction. It is embodied in VVC as BDOF to refine the prediction of the CU-based bi-directional inter prediction, and as PROF to refine the subblock prediction of AMC.

A. Bi-Directional Optical Flow for CU-Based Inter Prediction

Conventional bi-prediction is a weighted combination of two prediction blocks from previously coded pictures. Two MVs are used to obtain the two prediction blocks from a list 0 reference picture and a list 1 reference picture, respectively. However, due to the limitation of block-based MC, there are usually remaining displacements between the samples inside the two prediction blocks. The BDOF aims at compensating such fine displacement for each prediction sample on top of the original block-level MVs. In opposite to the block-based motion compensation, the refinement values of MVs are not signaled in BDOF. The BDOF is applied to the luma CB of a CU in regular merge mode or in the inter mode where symmetric MVD is not applied. And the BDOF is restricted to CUs with width and height larger than or equal to 8 luma samples and the number of luma samples greater than or equal to 128. Other restrictions on the usage of BDOF can be found in [23].

The BDOF is built upon the optical flow concept [33], [34]. Let $I(i, j, t)$ be the luminance value of a sample at position (i, j) and time t. Assuming the luminance of each sample is constant during the movement of the object, the optical flow differential equation, in this case, can be expressed as follows

$$0 = \frac{\partial I}{\partial t} + v_x\frac{\partial I}{\partial x} + v_y\frac{\partial I}{\partial y}. \quad (19)$$

As shown in Fig. 8, at each sample position the motion (v_x, v_y) describing the remaining small displacement from I_c to I_0 is symmetrical to its motion from I_c to I_1. Here I_c, I_0 and I_1 are arrays of luminance values in the current block and

the two prediction blocks from the list 0 and list 1 reference pictures, respectively. For the simplicity, remaining motions relative to both reference pictures are assumed to be the same in magnitude and opposite in directions. This assumption is reflected by the constraint that BDOF is applied only if the two different reference pictures have equal distance in POC to the current picture. It is worth noting that the same constraint is applied to DMVR as well.

Based on the symmetric motion model, (19) can be used to approximate the value of each sample in I_c from two directions, one from its correspondence A in I_0 and the other from its correspondence B in I_1. The value of (v_x, v_y) is calculated by minimizing the difference between two predictions with refined motion:

$$(v_x, v_y) : \min \sum_{(i,j)\in\Omega} \Delta^2(i, j); \tag{20}$$

$$\begin{aligned}\Delta(i, j) = {} & I_0(i, j) - I_1(i, j) \\ & + v_x\left(\frac{\partial I_0(i, j)}{\partial x} + \frac{\partial I_1(i, j)}{\partial x}\right) \\ & + v_y\left(\frac{\partial I_0(i, j)}{\partial y} + \frac{\partial I_1(i, j)}{\partial y}\right).\end{aligned} \tag{21}$$

Here (i, j) is the spatial coordinate of a sample inside the predicted block, Ω is surrounding of this sample. Spatial derivatives are approximated for the discrete sample arrays $I_0(i, j)$ and $I_1(i, j)$:

$$\frac{\partial I_{0,1}(i, j)}{\partial x} = (I_{0,1}(i + 1, j) - I_{0,1}(i - 1, j)) \gg 1, \tag{22}$$

$$\frac{\partial I_{0,1}(i, j)}{\partial y} = (I_{0,1}(i, j + 1) - I_{0,1}(i, j - 1)) \gg 1. \tag{23}$$

To reduce the computational complexity of the derivation of local remaining motions, the vector (v_x, v_y) is assumed constant inside each 4×4 subblock. It is calculated once and shared by all the samples in the subblock [50]. Additionally, to make the derived motion field more stable, (v_x, v_y) of each 4×4 subblock is calculated from the extended 6×6 region (noted as Ω in (20)) containing a 4×4 subblock in the center.

The optimization problem in (20) can be resolved by setting both partial derivatives to zero and the resulting linear equation system is approximately solved by

$$v_x = -\frac{S_4}{S_1}, \quad v_y = -\frac{S_5 + v_x S_3}{S_2}, \tag{24}$$

where

$$S_1 = \sum_{(i,j)\in\Omega} \vartheta_x(i, j) \cdot \vartheta_x(i, j), \tag{25}$$

$$S_2 = \sum_{(i,j)\in\Omega} \vartheta_y(i, j) \cdot \vartheta_y(i, j), \tag{26}$$

$$S_3 = \sum_{(i,j)\in\Omega} \vartheta_x(i, j) \cdot \vartheta_y(i, j), \tag{27}$$

$$S_4 = \sum_{(i,j)\in\Omega} \vartheta_t(i, j) \cdot \vartheta_x(i, j), \tag{28}$$

and

$$S_5 = \sum_{(i,j)\in\Omega} \vartheta_t(i, j) \cdot \vartheta_y(i, j) \tag{29}$$

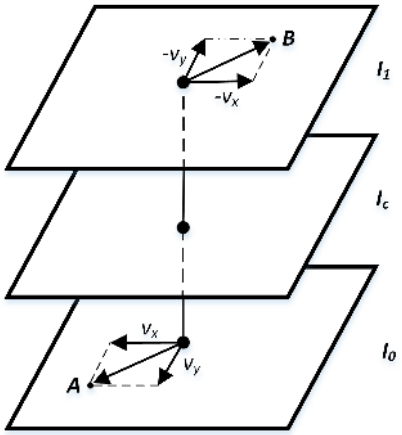


Fig. 8. Illustration of the symmetric motion model used by the BDOF (adapted from [34], © 2010 IEEE).

are the auto- and cross-correlation parameters, and $\vartheta_x(i, j)$ and $\vartheta_y(i, j)$ are the horizontal and vertical gradients and $\vartheta_t(i, j)$ is the temporal gradient of the sample at position (i, j), which are calculated as

$$\vartheta_x(i, j) = \frac{\partial I_0(i, j)}{\partial x} + \frac{\partial I_1(i, j)}{\partial x}, \tag{30}$$

$$\vartheta_y(i, j) = \frac{\partial I_0(i, j)}{\partial y} + \frac{\partial I_1(i, j)}{\partial y}, \tag{31}$$

and

$$\vartheta_t(i, j) = I_0(i, j) - I_1(i, j). \tag{32}$$

After (v_x, v_y) is derived, the final bi-prediction signal $I'_c(i, j)$ in the current position (i, j) of the block is calculated by interpolating list 0 and list 1 prediction samples along the motion trajectory (as shown in Fig. 8) based on the Hermite interpolation, i.e.,

$$I'_c(i, j) = \frac{1}{2}(I_0(i, j) + I_1(i, j) + \sigma_{\text{BDOF}}), \tag{33}$$

$$\begin{aligned}\sigma_{\text{BDOF}} = {} & v_x\left(\frac{\partial I_0(i, j)}{\partial x} - \frac{\partial I_1(i, j)}{\partial x}\right) \\ & + v_y\left(\frac{\partial I_0(i, j)}{\partial y} - \frac{\partial I_1(i, j)}{\partial y}\right).\end{aligned} \tag{34}$$

B. Affine Prediction Refinement With Optical Flow

At a later stage of VVC standardization, PROF was adopted to compensate for the prediction error of subblock-based AMC by applying the optical flow-based sample-wise refinement to the prediction.

Let P be the affine subblock prediction. Assume $P(i, j)$, the prediction at position (i, j) in the current subblock is predicted from sample $I(x, y)$ at position (x, y) in the reference picture with the subblock MV. Let (u_{ij}, v_{ij}) be the displacement between the sample MV and the subblock MV, $I(x + u_{ij}, y + v_{ij})$ would be the prediction if sample MV is used for motion compensation. The subblock-based and sample-based AMC are illustrated in Fig. 9.

Assuming (u_{ij}, v_{ij}) is small, $I(x + u_{ij}, y + v_{ij})$ can be approximated by Taylor expansion:

$$I(x + u_{ij}, y + v_{ij}) \approx I(x, y) + \frac{\partial I}{\partial x} u_{ij} + \frac{\partial I}{\partial y} v_{ij}. \quad (35)$$

The prediction refinement is obtained by

$$\Delta I = g_x(i, j) u_{ij} + g_y(i, j) v_{ij}, \quad (36)$$

where $g_x(i, j)$ and $g_y(i, j)$ are the spatial gradients of the subblock prediction in the horizontal and vertical directions.

The gradient calculation is performed for each 4×4 subblock in an affine-coded CU in the following steps. At first, the 4×4 subblock prediction is extended by one sample on each side. The extended samples are copied from the nearest integer positions in the reference picture to avoid additional memory bandwidth consumption. Then, for each position (i, j) in the 4×4 subblock, the spatial gradient is calculated as follow:

$$g_x(i, j) = (P(i + 1, j) \gg 6) - (P(i - 1, j) \gg 6), \quad (37)$$

$$g_y(i, j) = (P(i, j + 1) \gg 6) - (P(i, j - 1) \gg 6). \quad (38)$$

This is equivalent to applying a 2-tap filter on prediction samples for gradient calculation.

The displacement (u_{ij}, v_{ij}) associated to each sample in the a subblock of an affine-coded CU can be derived based on the affine matrix A of the CU and the sample position relative to the center of a subblock with the following equation:

$$(u_{ij}, v_{ij}) = (i - 1.5, j - 1.5) \times A, \quad (39)$$

where (1.5, 1.5) is the center position relative to the top-left sample in a subblock. A is the affine matrix of the CU:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad (40)$$

where a, b, c, and d are the affine model parameters as in (1). Since the sample positions relative to the subblock center are independent of the subblock position in a CU, (u_{ij}, v_{ij}) can be derived for one subblock and reused in all subblocks in the same CU in an optimized implementation.

The intermediate precision of the calculation is carefully designed to reduce the rounding error. The precision of the spatial gradient in (37)–(38) is designed to be the same as that in the BDOF process (see Section IV-C.3), such that the gradient computation component can be reused. u_{ij} and v_{ij} are rounded to $1/32$ fractional-sample accuracy and further clipped within the range of $[-31/32, 31/32]$ to avoid large MV difference.

The prediction refinement $\Delta I(i, j)$ is then clipped according to the internal bit-depth and added to the affine subblock prediction to form the final affine prediction. PROF is always applied to CUs in affine mode, but bypassed in case of identical CPMVs of an affine model, reference picture resampling [1], [51] being used, or the affine fallback mode (see Section IV-A.3) being triggered. The affine fallback mode will be triggered when subblocks of an affine-coded CU spread far apart from each other which poses a challenge to the memory bandwidth consumption during MC.

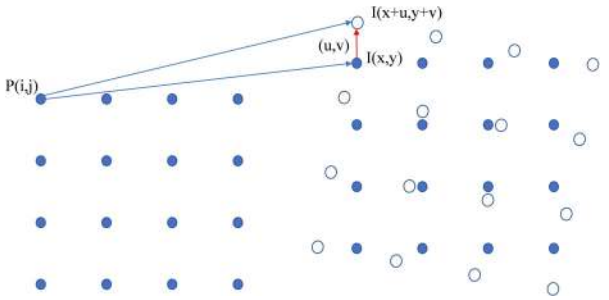


Fig. 9. Subblock-based AMC and sample-based AMC.

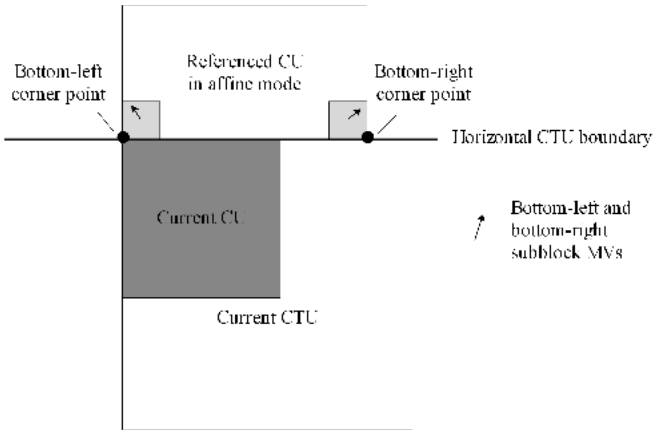


Fig. 10. Affine model inheritance from above CTU.

IV. Implementation Considerations

The preliminary designs of AMVC, DMVR and BDOF have been proposed and studied inside the JVET for a long time, but have not been considered mature until they were simplified in several important aspects for feasible hardware implementations with an affordable cost. Therefore, important simplifications of the three tools are elaborated in this section.

A. Affine Motion Compensation

1) Motion Information Storage of Affine-Coded CUs: After processing an affine-coded CU, the motion information of all 4×4 subblocks of the luma CB is stored and will be used in subsequent processes such as the MV prediction of following CUs and the deblocking filtering on block boundaries. Due to the introduction of the model inheritance mechanism which refers to the affine model of a previous CU coded in affine mode, the CPMVs of an affine-coded CU need to be additionally stored in a separate buffer [17]. Specifically, the stored CPMVs of neighboring CUs are used to derive the inherited affine MVP candidates and the inherited affine merge candidates for the current CU.

2) Affine Model Inheritance From the Above CTU Line: In a typical hardware design, a line buffer is used to store the motion information of the above CTU line and then the CUs in the current CTU line can refer to the stored motion information for MV prediction and deblocking filtering. Similarly, the CPMVs of the affine-coded CUs located at the bottom of the above CTU line need to be additionally stored and then

can be used by the affine-coded CUs located at the top of the current CTU line, according to the model inheritance mechanism. To avoid adding additional line buffer for CPMVs in the above CTU line, a modified model inheritance mechanism was adopted in VVC to handle the situation. Specifically, when an affine-coded CU at the bottom of the above CTU line is referred to, the bottom-left and bottom-right subblock MVs of the CU are fetched and treated as the MVs at the bottom-left and bottom-right corner points, and a 4-parameter affine model represented by the two corner point MVs is assumed for the CU. The model is then applied to derive the inherited affine MVP candidates and the inherited affine merge candidates for the current CU [18], as shown in Fig. 10. With such a design, the subblock MVs already stored in the line buffer is reused for affine model inheritance.

3) Bounding Box Constraint for Low Complexity MC: The reference blocks for the MC of all 4×4 subblocks of an affine-coded CU may spread to a large region in the reference picture and have zero overlapping with each other, which may lead to non-affordable peak memory bandwidth consumption when fetching reference samples for MC [21]. To constrain the complexity of MC in such a case, the MV at the center position $(W/2, H/2)$ of the CU is calculated according to the affine model and is set to be the MVs of all subblocks. This is called affine fallback mode and is triggered if the size of the bounding box calculated for an affined-coded CU exceeds a pre-defined threshold. The bounding box is defined as the rectangular region covering all reference blocks of a pre-defined cluster of subblocks in the current CU. A reference block contains all reference luma samples required for the MC of the corresponding subblock.

For an affine-coded CU with bi-prediction, the bounding box covers all four reference blocks of a 2×2 cluster of subblocks, as shown in Fig. 11. The size of the bounding box is $W_{box} \times H_{box}$. W_{box} and H_{box} are calculated as,

$$W_{box} = \max(0, 4(1+a), 4c, 4(1+a)+4c) - \min(0, 4(1+a), 4c, 4(1+a)+4c) + 9, \quad (41)$$

$$H_{box} = \max(0, 4b, 4(1+d), 4b+4(1+d)) - \min(0, 4b, 4(1+d), 4b+4(1+d)) + 9. \quad (42)$$

where a, b, c, and d are the affine parameters. The pre-defined threshold is set to $225 = (8+7) \times (8+7)$, which is the number of luma reference samples required for 8×8 block MC with 8-tap DCT-IF. By this design, the memory bandwidth consumption does not exceed that for the bi-prediction of inter-coded 8×8 CU in VVC.

Similarly, for an affine-coded CU with uni-prediction, the bounding box covers the two reference blocks of the 2×1 or 1×2 cluster of subblocks. The threshold is set to $165 = (8+7) \times (4+7)$ for both bounding boxes, where 165 is the number of luma reference samples required for the MC of an 8×4 or a 4×8 block. And the memory bandwidth consumption does not exceed that for the uni-prediction of inter-coded 8×4 or 4×8 CU in VVC by this design.

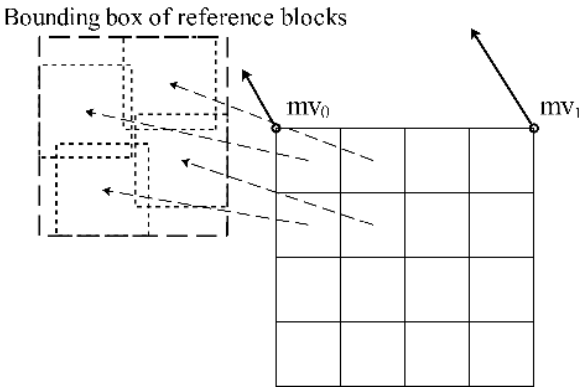


Fig. 11. Bounding box for a 2×2 cluster of subblocks.

B. Decoder-Side Motion Vector Refinement

The predecessors of DMVR such as FRUC with BM and TM in JEM have significantly high decoding complexity. A series of simplifications were proposed for DMVR to arrive at the sweet spot between compression gains and decoding/encoding complexity. The usage of bilinear interpolation during refinement, row subsampled SAD during the integer-sample search, and parametric error surface-based fractional-sample search are such simplifications already covered in Section II-C. Additional important simplifications are presented here.

1) DMVR Search Range and Integer-Sample Search Precision: The main source of the computational complexity of DMVR is SAD cost calculations in the integer-sample search step, which is proportional to the number of candidate MV pairs. The candidate MVs, either list 0 or list 1 MVs of the candidate MV pairs, point to search points that form a grid with a displacement of one luma sample in horizontal and vertical directions. The selection of the integer-sample interval as the displacement is a deliberate design choice, since the generation of predictions $P0_{i,j}$ and $P1_{i,j}$ by interpolation filtering is greatly simplified due to the following relationships:

$$P0_{i,j}[m-i, n-j] = P0_{0,0}[m, n], \quad (43)$$

$$P1_{i,j}[m-i, n-j] = P1_{0,0}[m, n]. \quad (44)$$

According to the above relationships the 25 prediction pairs of size W and H can be computed as a single prediction pair with size $W+2\mathcal{R}$ and $H+2\mathcal{R}$, where $\mathcal{R}$ is the search range.

The search range, which determines the number of equally spaced candidate MVs in the integer-sample search step, is set equal to 2 luma samples for the best trade-off of computational complexity and coding efficiency. This results in the aforementioned 25 candidate MV pairs and 25 necessary SAD computations.

2) Forced Subblock Partition: The motion refinement is performed for the entire luma CB in an early-phase design of DMVR. Performing refinement at luma block sizes of 128×128 requires two internal buffers of size up to $(128+4) \times (128+4)$, which is quite prohibitive. Also, handling the granularity of the different luma CB sizes while performing the refinement poses additional complexity. For these reasons,

the luma CB is partitioned into subblocks with luma width and height not exceeding 16. The refinement is performed independently for each subblock. Different subblock maximum sizes were experimented in [31]. The subblock partition design lowers the internal memory requirement in hardware to as low as $2 \times (16+7) \times (16+7)$ samples. Also, the refinement logic needs to handle only three possible subblock sizes, namely, 16×8, 8×16, and 16×16. It is noted that this forced subblock partition increases the overall number of operations required for MC in the final stage, compared to luma CB-based refinement. Since hardware decoders are designed to handle the theoretical worst-case number of partitions, this aspect is more of a concern only for software implementations.

3) Restriction on Using Refined MV: In typical hardware architectures of the MC module, reference samples are pre-fetched at a granularity higher than that of a CU through direct memory access (DMA). In a preliminary design of DMVR, the refined MVs are used as spatial MV predictors for subsequent CUs. This complicates the existing pre-fetching mechanism since the refined MVs inside a region are not available at the time when pre-fetching is working on the region. On the other hand, the derivation of the MVs of the subsequent CUs may need to wait for the refined MV of the current CU, which potentially adds additional latency for typical hardware pipelines. To eliminate potential implementation difficulties mentioned above, the use of refined MVs for merge MV derivation and spatial MV prediction was disabled [52]. Later, in [31], the use of refined MVs in boundary strength (BS) derivation for deblocking filtering was also disabled. Hence in VVC the refined MVs are used only for the motion compensation for the current CU and are stored in temporal MV buffer for coding of subsequent pictures. It is worth noting that the same restriction is applied to the BDOF process, where the refined motions of all 4×4 subblocks are used exclusively for the prediction refinement of the current CU.

4) Sample Padding for MC in the Final Stage: The conventional MC for a $W \times H$ block requires fetching up to $(W+7) \times (H+7)$ luma reference samples and $(W/2+3) \times (H/2+3)$ chroma reference samples per reference picture. With the ± 2 search window around the starting point for DMVR, $(W+11) \times (H+11)$ luma and $(W/2+5) \times (H/2+5)$ chroma reference samples would need pre-fetching. Considering the worst-case situation where each block would need to access the memory independently for its reference samples, the memory bandwidth consumption would be increased by DMVR. To maintain the same worst-case memory bandwidth, the block of reference samples is constrained to be the same as that for the conventional MC. In case samples outside of the reference area are required for MC in DMVR, padded samples are used instead. The padded sample values are obtained from the closest sample positions falling within the block of reference samples for conventional MC [53].

C. Bi-Directional Optical Flow

1) Forced Subblock Partition and Subblock-Based Operations: Following the same logic of subblock partitioning in the DMVR process, when the width and/or height of a luma CB are larger than 16 luma samples, it will be split into subblocks with width and height not exceeding 16. With such a design, the internal buffer size required by BDOF implementations is reduced, and the unit size for BDOF processing is perfectly aligned with that for DMVR processing in hardware codec design. Meanwhile, as will be discussed later, such a design also allows the initial SAD of the DMVR refinement to be used for bypassing partial BDOF operations. Without loss of generality, 16×16 size subblock is assumed for the explanation hereinafter in this section.

TABLE I

BIT-WIDTHS OF INTERNAL BDOF PARAMETERS WHEN INTERNAL BIT-DEPTH RANGING FROM 8 TO 16 BIT. TOP: RANGE, BOTTOM: BIT-WIDTH

Parameter	$8 \sim 12$ bit	14 bit	16 bit
S_1, S_2	$[0, 28224]$	$[0, 112896]$	$[0, 451584]$
S_3	$[-28224, 28224]$	$[-112896, 112896]$	$[-451584, 451584]$
S_4, S_5	$[-112860, 112860]$	$[-451548, 451548]$	$[-1806300, 1806300]$
v_x, v_y	$[-16, 15]$	$[-16, 15]$	$[-16, 15]$
σ_{BDOF}	$[-50144, 48576]$	$[-50144, 48576]$	$[-802784, 777696]$

Parameter	$8 \sim 12$ bit	14 bit	16 bit
S_1, S_2	16	18	20
S_3	16	18	20
S_4, S_5	18	20	22
v_x, v_y	5	5	5
σ_{BDOF}	17	19	21

The gradient calculations specified in (22)–(23) are conducted at the 16×16 subblock level. Each 16×16 subblock is extended by one sample on each side. Instead of performing motion-compensated interpolation to obtain the extended samples, these samples are copied from the nearest integer sample positions in the reference picture to avoid additional memory bandwidth and additional MC operations. The obtained gradients are then used for the calculation of v_x, v_y, and $I'_c(i, j)$ as specified in (24)–(34), which is conducted for each 4×4 subblock inside a 16×16 subblock. As each 4×4 subblock is extended to a 6×6 region for the calculations, prediction samples and gradients outside of the current 16×16 subblock boundaries are requested for 4×4 subblocks located at the boundaries of the 16×16 subblock. These prediction samples and gradients are directly copied from their nearest neighbors on the 16×16 subblock boundaries [37].

An early termination for BDOF operations is applied at the 16×16 subblock level [36]. Specifically, the BDOF process may be skipped for a subblock when the DMVR is applied to the subblock as well. In this case, the SAD cost with the initial MV pair from the DMVR process is checked. If the SAD cost is smaller than a threshold, the prediction is considered to be of high quality, and therefore the BDOF process is skipped. The threshold is set equal to $(2 \times W \times H)$, where W and H indicate the width and height of the subblock.

2) Simplified BDOF Parameter Derivation: As shown in (25)–(29), large numbers of multiplications are needed to calculate the auto- and cross-correlation parameters $S1, S2, \ldots, S5$ in a straightforward design. This results in

TABLE II
CODING PERFORMANCE AND RELATIVE RUN TIMES OVER ALL PRESENTED CODING TOOLS

Class	Sequence	RA (%)						LDB (%)					
		Y	U	V	YUV	EncT	DecT	Y	U	V	YUV	EncT	DecT
Class A1	*Tango2*	6.1	5.6	5.2	5.9	134	122	/	/	/	/	/	/
	FoodMarket4	7.2	6.6	6.3	7.0	142	131	/	/	/	/	/	/
	Campfire	0.6	0.2	0.4	0.5	116	107	/	/	/	/	/	/
Class A2	*CatRobot1*	14.2	11.0	10.1	13.4	136	120	/	/	/	/	/	/
	DaylightRoad2	11.5	8.6	8.6	10.8	137	123	/	/	/	/	/	/
	ParkRunning3	5.6	3.7	3.9	5.3	125	120	/	/	/	/	/	/
Class B	*MarketPlace*	7.6	5.9	6.3	7.3	133	125	3.9	2.6	2.8	3.7	144	126
	RitualDance	4.5	3.5	3.3	4.3	134	120	1.8	1.5	1.6	1.7	140	114
	Cactus	11.5	9.5	8.3	11.0	128	119	10.9	0.1	10.4	10.8	144	112
	BasketballDrive	3.3	3.0	2.8	3.2	127	118	1.9	2.3	1.3	1.9	136	110
	BQTerrace	2.2	2.0	1.8	2.1	133	116	0.6	0.2	0.5	0.5	142	101
Class C	*BasketballDrill*	2.7	2.0	2.1	2.6	126	121	1.1	1.0	0.7	1.1	131	100
	BQMall	5.4	4.5	4.0	5.2	126	127	2.1	1.1	1.6	2.0	132	103
	PartyScene	4.9	4.2	3.8	4.7	125	125	5.5	5.0	4.5	5.4	129	116
	RaceHorses	2.5	1.9	1.9	2.4	128	116	1.3	0.6	1.2	1.2	125	99
Class E	*FourPeople*	/	/	/	/	/	/	3.5	2.9	4.2	3.5	154	107
	Johnny	/	/	/	/	/	/	4.5	4.6	5.5	4.7	171	110
	KristenAndSara	/	/	/	/	/	/	4.7	4.4	5.1	4.7	162	111
Average	Class A1	4.6	4.1	4.0	4.4	130	120	/	/	/	/	/	/
	Class A2	10.4	7.7	7.5	9.8	132	121	/	/	/	/	/	/
	Class B	5.8	4.8	4.5	5.6	131	120	3.8	3.3	3.3	3.7	141	112
	Class C	3.9	3.2	3.0	3.7	126	122	2.5	1.9	2.0	2.4	129	104
	Class E	/	/	/	/	/	/	4.3	4.0	5.0	4.3	162	110
	Overall	**6.0**	**4.8**	**4.6**	**5.7**	**130**	**121**	**3.5**	**3.0**	**3.3**	**3.4**	**142**	**109**

$5 \times 6 \times 6$ multiplications for each 4×4 subblock, as the motion refinements are derived in the extended 6×6 region. Based on the assumption that gradients are nearly constant inside a subblock, the multiplications can be replaced by sign operations and (25)–(29) are approximated by:

$$S_1 = \sum_{(i,j)\in\Omega} \text{abs}(\vartheta_x(i,j)), \tag{45}$$

$$S_2 = \sum_{(i,j)\in\Omega} \text{abs}(\vartheta_y(i,j)), \tag{46}$$

$$S_3 = \sum_{(i,j)\in\Omega} \vartheta_x(i,j)\cdot \text{sign}(\vartheta_y(i,j)), \tag{47}$$

$$S_4 = \sum_{(i,j)\in\Omega} \vartheta_t(i,j)\cdot \text{sign}(\vartheta_x(i,j)), \tag{48}$$

and

$$S_5 = \sum_{(i,j)\in\Omega} \vartheta_t(i,j)\cdot \text{sign}(\vartheta_y(i,j)). \tag{49}$$

This approximation removes all multiplications and reduces the overall number of multiplications of BDOF by more than 80 % [38].

3) Bit-Width Control of BDOF Intermediate Parameters: To reduce the dynamic range of intermediate parameters used for the BDOF process, different bitwise right shifts are introduced [37] to lower the precisions of the BDOF parameters $\vartheta_x(i,j)$, $\vartheta_y(i,j)$ and $\vartheta_t(i,j)$ in (30)–(32), as indicated below:

$$\vartheta_x(i,j) = (g_x^0(i,j) + g_x^1(i,j)) \gg 1, \tag{50}$$

$$\vartheta_y(i,j) = (g_y^0(i,j) + g_y^1(i,j)) \gg 1, \tag{51}$$

and

$$\vartheta_t(i,j) = (I_0(i,j) \gg 4) - (I_1(i,j) \gg 4), \tag{52}$$

where

$$g_x^{0,1}(i,j) = (I_{0,1}(i+1,j) \gg 6) - (I_{0,1}(i-1,j) \gg 6), \tag{53}$$

$$g_y^{0,1}(i,j) = (I_{0,1}(i,j+1) \gg 6) - (I_{0,1}(i,j-1) \gg 6). \tag{54}$$

Table I illustrates the bit-widths of the BDOF intermediate parameters in (21)–(34) for different internal bit-depths. As shown in the table, for internal bit-depths from 8 to 16 bits, all the BDOF related computations can be implemented using integer arithmetic not exceeding 32 bits.

V. EXPERIMENTAL RESULTS AND ANALYSIS

Encoder configurations of random access (RA) and low delay with B slices (LDB) specified in the JVET common test conditions (CTC) [54] were used for evaluating the implementations of the five subblock-based inter coding tools in VTM-9.0 [55]. In CTC, test sequences are grouped into several categories according to the spatial resolution and the application scenario. Class A1 and A2 contain 4K sequences representing the high quality entertainment video content and therefore they are not tested with LDB configuration that is primarily for real-time communication scenario. Out of the same reason, class E that comprises three typical video conferencing sequences is tested with LDB only. The Bjøntegaard Delta rate (BD-rate) [56] was employed to measure the bitrate

TABLE III
CODING GAIN AND RELATIVE RUN TIMES WHEN SWITCHING ON INDIVIDUAL TOOL

Tool On	AMC	AMC +PROF	SbTMVP	DMVR	BDOF
Class A1	2.3 %	2.4 %	0.8 %	1.5 %	0.9 %
Class A2	6.2 %	6.8 %	0.9 %	3.0 %	2.3 %
Class B	3.1 %	3.6 %	0.7 %	1.5 %	1.4 %
Class C	1.2 %	1.5 %	0.7 %	1.1 %	1.2 %
Average	**3.0 %**	**3.4 %**	**0.8 %**	**1.7 %**	**1.4 %**
EncT	127 %	129 %	104 %	101 %	104 %
DecT	108 %	111 %	105 %	112 %	111 %

TABLE IV
CODING GAIN AND RELATIVE RUN TIMES WHEN SWITCHING OFF INDIVIDUAL TOOL

Tool Off	PROF	AMC +PROF	SbTMVP	DMVR	BDOF
Class A1	−0.1 %	−2.2 %	−0.5 %	−0.8 %	−0.4 %
Class A2	−0.9 %	−6.1 %	−0.4 %	−1.4 %	−1.0 %
Class B	−0.6 %	−3.2 %	−0.3 %	−0.7 %	−0.7 %
Class C	−0.3 %	−1.4 %	−0.5 %	−0.6 %	−0.8 %
Average	**−0.5 %**	**−3.1 %**	**−0.4 %**	**−0.8 %**	**−0.8 %**
EncT	99 %	83 %	101 %	100 %	97 %
DecT	100 %	98 %	101 %	98 %	98 %

savings with the same reproduction quality. The reproduction quality of a test sequence is measured by PSNR for three color components of Y, U, and V separately or jointly. Weighted average of the PSNR of Y, U, and V components with 6 : 1 : 1 ratio [57] was used to measure the quality of the three color components jointly. For each test sequence, four quantization parameter (QP) values 22, 27, 32, and 37 were used to generate four rate points and the piece-wise cubic interpolation was used for BD-rate calculation [54].

A. Rate-Distortion Performance

The anchor for comparison is VTM-9.0 with the five tools switched off. The compression efficiency improvement in terms of BD-rate as well as the computational complexity increase in terms of the encoder and decoder runtime ratio are presented in Table II. Averaged BD-rate savings over all test sequences with joint YUV measurement are 5.7 % and 3.4 % with RA and LDB configurations, respectively. In addition, a balanced performance on Y, U, and V components can be observed. Since DMVR and BDOF are not applicable to the LDB configuration where the two reference pictures for bi-prediction come from the same temporal direction, the compression efficiency in LDB is lower than that in RA. With the RA configuration, more than 10 % BD-rate saving is observed for test sequence *CatRobot1*, *Cactus*, and *DaylightRoad2* where complex motion such as rotation and zooming are observed. It is worth noting that such complex motion typically consumes a lot of bits for coding with previous standards and therefore is considered to be very challenging for video compression. However, the five VVC tools presented in this paper are especially good at handling complex motion according to the results. Since the encoder and decoder runtime increase in percentage highly depends on the specific implementation and the level of optimization, the data provided in Table II can only be referenced as a rough estimation of the computational complexity. It is observed that the encoder runtime in LDB is higher than that in RA. This is caused by the specific implementation of AMC motion estimation module in VTM-9.0 where more reference pictures are checked in the bi-prediction stage with the LDB configuration.

The gain in compression efficiency with regard to individual tools are presented in Table III. Only the RA configuration is tested because DMVR and BDOF are not applicable to LDB. As shown in Table III, all tools show decent gain while AMC shows the biggest gain. And in general the overall complexity considering both encoder and decoder runtime correlates well with the BD-rate saving for the five tools. Therefore all the tools demonstrate good trade-off between the compression efficiency and the computational complexity. Note that PROF is only applicable to affine-coded CUs and therefore can only be tested on top of AMC.

Different from Table II and Table III, VTM-9.0 with all five tools switched on was used as the anchor in Table IV, and each tool was switched off individually to observe the compression efficiency loss, where the negative sign of a BD-rate value indicates a loss. Averaged PSNR of Y, U, and V components is used for the BD-rate calculation in Table III and Table IV. We can get a hint of the interaction of the five tools by checking the corresponding BD-rate values in Table III and Table IV. For example, the BD-rate of AMC + PROF is similar to that in Table III in absolute value but different in sign, which shows that AMC + PROF has almost no overlap with the other three tools in terms of compression efficiency gain. However, for SbTMVP, DMVR and BDOF, the absolute values of BD-rate in Table IV are about half of the corresponding ones in Table III, indicating that each tool has a certain level of overlap with the set of the remaining four tools. Since all the three tools try to save the signaling overhead of motion information, it is reasonable to observe an overlap with each other because of competition. Furthermore, each of the three tools may have an overlap with AMC which tries to reduce the signaling overhead of motion information by a more compact representation. However, mild overlap actually gives more flexibility in practical applications. Assuming that a certain implementation of the VVC encoder may not favor a specific coding tool, in this case, this tool can be switched off while the potential compression efficiency loss can be partially compensated by other tools.

B. Statistics Analysis

Table V shows the percentage of bitrate reduction for the test of Class A2 sequences with RA configuration. Class A2 was selected for illustration because it shows top performance among all classes and therefore can better demonstrate the trends in statistics. We observed that the five tools can significantly reduce the bits for coding the partition and motion

TABLE V
STATISTICS OF THE BITRATE REDUCTION FOR CLASS A2 WITH RA

QP	Total	Partition	Motion	Residual
22	2.1 %	0.4 %	1.3 %	0.3 %
27	5.2 %	0.9 %	3.3 %	0.5 %
32	6.8 %	1.2 %	4.6 %	0.4 %
37	7.6 %	1.3 %	5.3 %	0.3 %

(a) Tools disabled (b) Tools enabled

Fig. 12. CU partitioning of the 9^{th} frame of *CatRobot1*.

information. For example, in case of QP 37, the bits reduction for coding partition information takes 1.3 % of the total bitstream, and the bits reduction for coding motion information is 5.3 %. This is because the five tools are designed to either represent the motion information in a more efficient way or to derive the motion at the decoder side. And therefore the bits for coding fine-granular motion in small blocks can be saved. The bits for coding prediction residual can also be reduced because more accurate prediction can be achieved by the five tools. It is also observed that the total bitrate saving increased dramatically with QP values, reaching up to 7.6 % for QP 37. The reason is that the portion of partition and motion information in the bitstream becomes larger for higher QP values, and therefore the advantages of the five tools become more evident.

Fig. 12(a) and Fig. 12(b) show the CU partitioning in a part of the 9^{th} frame of the sequence *CatRobot1*, with the five tools disabled and enabled. It can be obviously seen that the CU size increases a lot in the area of the rotating plate. Without the five subblock-based inter tools, the encoder has to split the complex motion region into small CUs and transmit huge amount of motion information. With the five tools enabled, the complex motion can be efficiently represented and further refined at the decoder side, and therefore large CU sizes are selected instead.

C. Visual Quality Impact

The reconstruction for a part of the 41^{st} frame of *CatRobot1*, with the five tools disabled and enabled, is shown in Fig. 13. RA configuration was used for the simulation. With the five tools enabled, the bits for coding the frame is decreased to 72928 from 90088. Meanwhile, a much better visual quality is observed. Specifically, the blocking artifact along the sharp edges on the rotating plate is largely removed. We believe that the visual quality can be further improved if

(a) Tools disabled (b) Tools enabled

Fig. 13. Subjective quality of the 41^{st} frame of *CatRobot1*.

the bits for coding the two frames are aligned. In the example in Fig. 13, most of the visual quality benefits are contributed by AMC and PROF since the two tools are especially good at handling rotations in video content. For the other three tools, SbTMVP, DMVR and BDOF, it is observed that their primary effect is reducing the bits for coding motion information rather than improving the quality of reconstruction.

VI. CONCLUSION

Five VVC inter coding tools, AMC, DMVR, SbTMVP, BDOF and PROF, are introduced in this paper. These tools are designed to perform fine-granular motion compensation without explicit signaling of subblock MVs. Design elements considering typical hardware implementations are presented as well. Experiments are conducted to demonstrate the advantages in RD efficiency and to provide a comprehensive analysis of the inherent characteristics of the tools.

ACKNOWLEDGMENT

The authors would like to thank Han Gao who contributed to the drafting of this paper. They thank Prof. Shuai Wan for the help to improve the English writing of this paper. They thank JVET experts for their great contributions to the design of the subblock-based inter coding tools throughout the standardization period of VVC.

REFERENCES

[1] B. Bross *et al.*, "Overview of the versatile video coding (VVC) standard and its applications," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.
[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[3] P. Helle *et al.*, "Block merging for quadtree-based partitioning in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1720–1731, Dec. 2012.
[4] J.-L. Lin, Y.-W. Chen, Y.-W. Huang, and S.-M. Lei, "Motion vector coding in the HEVC standard," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 957–968, Dec. 2013.
[5] W.-J. Chien *et al.*, "Motion vector coding and block merging in versatile video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, to be published, doi: 10.1109/TCSVT.2021.3101212.
[6] Y.-W. Huang *et al.*, "Block partitioning structure in the VVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Jun. 11, 2021, doi: 10.1109/TCSVT.2021.3088134.

[7] H. Huang, J. Woods, Y. Zhao, and H. Bai, "Control-point representation and differential coding affine-motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1651–1668, Oct. 2013.

[8] M. Narroschke and R. Swoboda, "Extending HEVC by an affine motion model," in *Proc. Picture Coding Symp. (PCS)*, Dec. 2013, pp. 321–324.

[9] C. Heithausen and J. H. Vorwerk, "Motion compensation with higher order motion models for HEVC," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 1438–1442.

[10] H. Chen, F. Liang, and S. Lin, "Affine SKIP and MERGE modes for video coding," in *Proc. 17th IEEE Int. Workshop Multimedia Signal Process. (MMSP)*, Oct. 2015, pp. 1–5.

[11] S. Lin, H. Chen, H. Zhang, M. Sychev, H. Yang, and J. Zhou, *Affine Transform Prediction for Next Generation Video Coding*, document ITU-T SG16/Q6, COM16-C1016, Oct. 2015.

[12] L. Li *et al.*, "An efficient four-parameter affine motion model for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 8, pp. 1934–1948, Aug. 2018.

[13] J. Chen, M. Karczewicz, Y.-W. Huang, K. Choi, J.-R. Ohm, and G. J. Sullivan, "The joint exploration model (JEM) for video compression with capability beyond HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1208–1225, May 2020.

[14] K. Zhang, Y.-W. Chen, L. Zhang, W.-J. Chien, and M. Karczewicz, "An improved framework of affine motion compensation in video coding," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1456–1469, Mar. 2019.

[15] N. Zhang, X. Fan, D. Zhao, and W. Gao, "Merge mode for deformable block motion information derivation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 11, pp. 2437–2449, Nov. 2017.

[16] Y. Han *et al.*, *CE4.1.3: Affine Motion Compensation Prediction*, Tech. Rep., Joint Video Experts Team (JVET), JVET-K0337, Jul. 2018.

[17] M. Zhou and B. Heng, *CE4-Related: A Clean Up for Affine Mode*, document Joint Video Experts Team (JVET), JVET-L0047, Oct. 2018.

[18] M. Zhou, *CE4: Test Results of CE4.1.11 on Line Buffer Reduction for Affine Mode*, document Joint Video Experts Team (JVET), JVET-L0045, Oct. 2018.

[19] A. Tamse, M. W. Park, and K. Choi, *CE2-Related: MV Derivation for Affine Chroma*, document Joint Video Experts Team (JVET), JVET-M0192, Jan. 2019.

[20] H. Chen, H. Yang, J. Chen, H. Huang, and W.-J. Chien, *CE4-Related: Combination of Affine Mode Clean Up and Line Buffer Reduction*, document Joint Video Experts Team (JVET), JVET-L0694, Oct. 2018.

[21] M. Zhou, *CE2: On Restriction of Memory Bandwidth Consumption of Affine Mode (CE2-4.8)*, document Joint Video Experts Team (JVET), JVET-N0068, Mar. 2019.

[22] J. Li, C.-W. Kuo, and C. S. Lim, *CE2: Using the Shorter-Tap Filter for 4x4 Sized Partitions (Test 2.4.6)*, document Joint Video Experts Team (JVET), JVET-N0196, Mar. 2019.

[23] *Versatile Video Coding*, document Rec. ITU-T. H.266 and ISO/IEC 23090-3, Version 1, 2020.

[24] T. Murakami and S. Saito, *Advanced B Skip Mode With Decoder-Side Motion Estimation*, document ITU-T SG16/Q6, 37th Meeting, VCEG-AK12, 2009.

[25] S. Klomp, M. Munderloh, Y. Vatis, and J. Ostermann, "Decoder-side block motion estimation for H.264/MPEG-4 AVC based video coding," in *Proc. IEEE Int. Symp. Circuits Syst.*, Taipei, Taiwan, May 2009, pp. 1641–1644.

[26] K. Sugimoto, M. Kobayashi, Y. Suzuki, S. Kato, and C. S. Boon, "Inter frame coding with template matching spatio-temporal prediction," in *Proc. Int. Conf. Image Process. (ICIP)*, Oct. 2004, pp. 465–468.

[27] S. Kamp, M. Evertz, and M. Wien, *Decoder Side Motion Vector Derivation*, document ITU-T SG16/Q6, 33th Meeting, VCEG-AG16, 2007.

[28] S. Kamp and M. Wien, "Decoder-side motion vector derivation for block-based video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1732–1745, Dec. 2012.

[29] J. Chen *et al.*, *Further Improvements to HMKTA-1.0*, document ITU-T SG16/Q6, 52th Meeting, VCEG-AZ07, Jun. 2015.

[30] X. Chen, J. An, and J. Zheng, *Decoder-Side Motion Vector Refinement Based on Bilateral Template Matching*, document Joint Video Experts Team (JVET), JVET-D0029, Oct. 2016.

[31] S. Sethuraman, *CE9: Results of DMVR Related Tests CE9.2.1 and CE9.2.2*, document Joint Video Experts Team (JVET), JVET-M0147, Jan. 2019.

[32] J. Chen *et al.*, *Coding Tools Investigation for Next Generation Video Coding*, document ITU-T SG16/Q6, COM16-C806, Feb. 2015.

[33] A. Alshin and E. Alshina, *Bi-Directional Optical Flow*, document Joint Collaborative Team on Video Coding (JCT-VC), JCTVC-C204, Oct. 2010.

[34] A. Alshin, E. Alshina, and T. Lee, "Bi-directional optical flow for improving motion compensation," in *Proc. 28th Picture Coding Symp.*, Dec. 2010, pp. 422–425.

[35] A. Alshin and E. Alshina, "Bi-directional pptical flow for future video codec," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2016, pp. 84–90.

[36] X. Xiu *et al.*, *CE9-Related: A Simplified Bi-Directional Optical Flow (BIO) Design Based on the Combination of CE9.5.2 and CE9.5.3*, document Joint Video Experts Team (JVET), JVET-K0485, Jul. 2018.

[37] X. Xiu, Y. He, and Y. Ye, *CE9-Related: Complexity Reduction and Bit-Width Control for Bi-Directional Optical Flow (BIO)*, Joint Video Experts Team (JVET), JVET-L0256, Jul. 2018.

[38] Y. Kato, T. Toma, and K. Abe, *Simplification of BDOF*, document Joint Video Experts Team (JVET), JVET-O0304, Jun. 2019.

[39] J. Luo and Y. He, *CE2-Related: Prediction Refinement With Optical Flow for Affine Mode*, document Joint Video Experts Team (JVET), JVET-N0236, Mar. 2019.

[40] Y. He and J. Luo, *CE4-2.1: Prediction Refinement With Optical Flow for Affine Mode*, document Joint Video Experts Team (JVET), JVET-O0070, Jun. 2019.

[41] H. Chen, H. Yang, and J. Chen, *CE4: Separate List for Sub-Block Merge Candidates (Test 4.2.8)*, document Joint Video Experts Team (JVET), JVET-L0369, Oct. 2018.

[42] X. Xiu and Y. He, *CE4-Related: One Simplified Design of Advanced Temporal Motion Vector Prediction (ATMVP)*, document Joint Video Experts Team (JVET), JVET-K0346, Jul. 2018.

[43] H. Gao, S. Esenlik, Z. Zhao, E. Steinbach, and J. Chen, "Low complexity decoder side motion vector refinement for VVC," in *Proc. Picture Coding Symp. (PCS)*, Ningbo, China, Nov. 2019, pp. 1–5.

[44] H. Gao, S. Esenlik, Z. Zhao, E. Steinbach, and J. Chen, "Decoder side motion vector refinement for versatile video coding," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process. (MMSP)*, Kuala Lumpur, Malaysia, Sep. 2019, pp. 1–6.

[45] H. Gao, H. Gao, X. Chen, S. Esenlik, J. Chen, and E. Steinbach, "Decoder-side motion vector refinement in VVC: Algorithm and hardware implementation considerations," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Nov. 9, 2020, doi: 10.1109/TCSVT.2020.3037024.

[46] K. Ugur *et al.*, "Motion compensated prediction and interpolation filter design in H.265/HEVC," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 946–956, Jul. 2013.

[47] S. Sethuraman, J. Raj, and S. Kotecha, *Decoder Side MV Refinement/Derivation With CTB-Level Concurrency and Other Normative Complexity Reduction Techniques*, document Joint Video Experts Team (JVET), JVET-K0041, Jul. 2018.

[48] C.-C. Chen, W.-J. Chien, and M. Karczewicz, *CE9.2.5/9.2.6: DMVR With Template-Free Bilateral Matching*, document Joint Video Experts Team (JVET), JVET-K0359, Jul. 2018.

[49] S. Sethuraman, *CE9: Test 9.2.6 (Combines CE9.2.15/9.2.16 With Elements of 9.1.4 and 9.2.5)*, document Joint Video Experts Team (JVET), JVET-L0173, Oct. 2018.

[50] H.-C. Chuang *et al.*, *EE-Related: A Block-Based Design for Bi-Directional Optical Flow (BIO)*, document Joint Video Experts Team (JVET), JVET-F0022, Mar. 2017.

[51] Y.-K. Wang *et al.*, "The high-level syntax of the versatile video coding (VVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Apr. 5, 2021, doi: 10.1109/TCSVT.2021.3070860.

[52] S. Esenlik *et al.*, *CE9: DMVR With Bilateral Matching (Test 2.9)*, document Joint Video Experts Team (JVET), JVET-K0217, Jul. 2018.

[53] S. Esenlik, I. Krasnov, Z. Zhao, and J. Chen, *Non-CE9: DMVR Without Intermediate Buffers and With Padding*, document Joint Video Experts Team (JVET), JVET-K0275, Jul. 2018.

[54] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, *JVET Common Test Conditions and Software Reference Configurations for SDR Video*, document Joint Video Experts Team (JVET), JVET-N1010, May 2019.

[55] *VVC Test Model Software*. Accessed: Jul. 29, 2021. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tags/VTM-9.0

[56] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD Curves*, document ITU-T SG16/Q6 (VCEG), VCEG-M33, Apr. 2001.

[57] J. Ström *et al.*, *Summary Information on BD-Rate Experiment Evaluation Practices*, document Joint Video Experts Team (JVET), JVET-R2016, Apr. 2020.

Haitao Yang (Member, IEEE) received the B.S. and Ph.D. degrees in electronic engineering from Xidian University, Xi'an, China, in 2004 and 2009, respectively. From July 2009 to June 2010, he was with The Hong Kong University of Science and Technology, as a Post-Doctoral Research Associate, where he worked on video compression. In October 2010, he joined Huawei Technologies Company Ltd., Shenzhen, China, where he has been actively participating and contributing to video compression standardization activities conducted by ITU-T Video Coding Experts Group and ISO/IEC Moving Pictures Experts Group, including Joint Collaborative Team on Video Coding (JCT-VC) and Joint Video Exploration Team (JVET) activities. He is currently an Expert Engineer with Huawei Media Coding Laboratory. His research interests include video compression, processing, and communication.

Huanbang Chen received the B.E. and M.E. degrees in communication engineering from Sun Yat-sen University, China, in 2012 and 2015, respectively. In August 2015, he joined Media Coding Technology Laboratory, Huawei Technologies, Shenzhen, China, where he has been actively involved in the development of various video coding standards, including the VVC standard under development in the Joint Video Experts Team (JVET). He has been the main contributor and developer of the affine motion compensation, which is one of the key features of VVC standard. His current research interests include image/video coding and processing.

Jianle Chen (Senior Member, IEEE) received the B.S. and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 2001 and 2006, respectively. He was formerly with Samsung Electronics, Qualcomm, and Huawei, USA, where he focuses on the research of video technologies. Since 2006, he has been actively involved in the development of various video coding standards, including the HEVC standard, its scalable, format range, and screen content coding extensions, and most recently, the VVC standard under development in the Joint Video Experts Team (JVET). He has also been a main developer of the recursive partitioning structure with large block size, which is one of the key features of HEVC standard and its potential successors. He is currently the Director of Multimedia Research and Development Group, Qualcomm, Inc., San Diego, CA, USA. His research interests include video coding and transmission, point cloud coding, AR/VR, and neural network compression. He was an Editor of the HEVC specification version 2 (the scalable HEVC (SHVC) text specification) and SHVC Test Model. For VVC, he has been the Lead Editor of the Joint Exploration Test Model (JEM) and VVC Test Model (VTM). He is an Editor of the VVC Text Specification.

Semih Esenlik received the B.S. degree in electrical and electronics engineering from Boğaziçi University, Istanbul, in 2008, and the M.S. degree in telecommunications engineering from the Technical University of Munich, in 2010. From 2010 to 2013, he worked as a Research Engineer at Multimedia Coding and Standardization Group, Panasonic Research and Development Center, Germany. From 2013 to 2016, he worked as a Technical Product Manager at Turkcell, Turkey. In 2016, he started working at Huawei Research and Development Center, Germany, as a Principal Research Engineer and shortly after became a Team Leader of Video Coding Standardization Team. He has actively participated in the development of HEVC, EVC, and VVC codecs, is among the main contributors of decoder side motion vector refinement and geometric partitioning mode technologies in VVC. He was the primary chair of core experiment on decoder side motion vector derivation and refinement during the development of VVC.

Sriram Sethuraman (Senior Member, IEEE) received the Ph.D. degree in ECE from Carnegie Mellon University. He has been a Senior Principal Scientist with the Prime Video Playback Team, Amazon, Bengaluru, India, since July 2019, leading efforts related to encoding optimization, video quality measurement, ML-based restoration, and next-generation video compression. Prior to joining Amazon, he was the CTO and Senior VP of Ittiam Systems Pvt. Ltd., Bengaluru, since 2012. During his 17-year tenure at Ittiam, he was the architect of its technologies and products in the fields of video compression, video communication, media broadcast, and computer vision and machine learning. Prior to joining Ittiam, he served as a Senior Member of Technical Staff at Sarnoff Corporation. He has been a part of MPEG-4, MPEG-7, and VVC standardization efforts. He has 34 issued patents (and several pending patents) and is the author of more than 35 publications.

Xiaoyu Xiu (Member, IEEE) received the B.S. and M.S. degrees from Beijing University of Technology, Beijing, China, in 2003 and 2006, respectively, both in electrical engineering, and the Ph.D. degree from the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada, in 2012.

From May 2010 to September 2010, he was a Visiting Researcher with the National Institute of Informatics, Tokyo, Japan. From 2012 to 2018, he was a Senior Staff Engineer at InterDigital Communications Inc. He is currently a Video Algorithm Scientist at Kwai Inc. He had been actively involved in various video coding standards, including the HEVC standard, it's scalable and screen content coding extensions and Joint Exploration Model of the JVET and the emerging VVC standard. During the development of those video coding standards, he co-chaired several *ad-hoc* groups and core experiments and software coordinator of the test model of the HEVC screen content coding extension. His research interests include 2D/3D image/video coding, video processing, streaming, and rate-distortion theory.

Elena Alshina received the M.S. degree in physics from Lomonosov Moscow State University (MSU) in 1995 and the Ph.D. degree in mathematical modeling from the Faculty of Computational Mathematics and Cybernetics, MSU, in 1998. She has been the Chief Video Scientist with Huawei, Munich, since May 2018. She is currently in charge of machine oriented video, AI-based codec development, and international standardization. Since that time, she has been working with the Institute for Mathematical Modeling, Russian Academy of Science, where she is performing research on high accuracy numerical methods and new finite-difference scheme development. Simultaneously, she worked as an Associate Professor with Moscow Institute of Electronic Technology. In 2006, she was with Multimedia Platform Laboratory, Digital Media and Communications Research and Development Center, Samsung. Since 2008, she has been an active participant of international standardization (MPEG/VCEG/JCTVC/JVET).

Jiancong Luo received the Ph.D. degree from the University of Texas at Arlington, TX, USA, in 2005. From 2005 to 2010, he was a Research Scientist with Technicolor, where he focused on the research of advanced video processing and compression technology, and contributed to the ISO/IEC MPEG AVC, SVC, and MVC standardization efforts. From 2010 to 2018, he was with Harmonic Inc., where he involved in the development of video codec systems. From 2018 to 2019, he was with InterDigital Inc., and Alibaba Inc., from 2019 to 2020, where he actively involved to the development of ISO/IEC MPEG VVC standard. He is currently with Apple Inc. His current research interests include high-efficiency compression and deep image/video processing technology.