VVC In-Loop Filters

Marta Karczewicz, Nan Hu, Jonathan Taquet, Ching-Yeh Chen, Kiran Misra, Kenneth Andersson, Peng Yin, Taoran Lu, Edouard François, and Jie Chen

(Invited Paper)

***Abstract*— This paper presents an overview of the technologies for in-loop processing and filtering in the Versatile Video Coding (VVC) standard. These processes comprise luma mapping with chroma scaling, deblocking filter, sample adaptive offset, adaptive loop filter and cross-component adaptive loop filter. They are qualified as "in-loop" because they are applied inside the encoding and decoding loops, before storing the pictures in the decoded picture buffer. The filters are complementary and address different purposes. Luma mapping with chroma scaling aims at adaptively modifying the coded samples distribution for improved coding efficiency. The deblocking filter aims at reducing blocking discontinuities. Sample adaptive offset mostly aims at reducing artifacts resulting from the quantization of transform coefficients. Adaptive loop filter and cross-component adaptive loop filter are adaptive filters enabling to enhance the reconstructed signal, using for instance Wiener-filter encoding approaches. The paper provides an overview of the in-loop filtering process and a detailed description of the filtering algorithms. Objective compression efficiency results are provided for each filter, with indication of cumulative coding gains. Subjective benefits are illustrated. Implementation issues considered during the design of the VVC in-loop filters are also discussed.**

***Index Terms*— Video coding, in-loop filters, adaptive loop filter, cross-component adaptive loop filter, deblocking, luma mapping with chroma scaling, Versatile video coding.**

I. Introduction

VERSATILE Video Coding (VVC) [1] is a new video coding standard developed by the Joint Video Experts Team (JVET) grouping experts from the ITU-T SG 16/Q.6 Video Coding Experts Group (VCEG), and the ISO/IEC JTC 1/SC 29/WG 11 Moving Pictures Experts Group (MPEG), which had also jointly developed the AVC (H.264) [2] and HEVC (H.265) [3] standards. As in previous video coding standards, VVC uses a block-based hybrid coding scheme, as illustrated in Fig. 1 that depicts a simplified VVC decoder block diagram emphasizing the in-loop filtering coding blocks (gray-shaded rectangles). The filters are defined as "in-loop" because these processes are applied inside the encoding/decoding loop that is prior to the picture storage in the decoded picture buffer (DPB). The decoding process starts with entropy decoding using a context-adaptive binary arithmetic coding (CABAC) engine, followed by inverse quantization and inverse transform that results in the decoded residue. The residue is added to the prediction signal (intra, inter, or mix of both in case of combined intra-inter prediction mode (CIIP)). The resulting reconstructed signal is then processed through different in-loop filtering steps. The filtered picture is finally stored in the DPB.

In VVC, the pictures are partitioned into Coding Tree Units (CTUs), which represent the basic coding processing units, also specified in HEVC. CTUs consist of one or three Coding Tree Blocks (CTBs) depending on whether the video signal is monochrome or contains three-color components. For YCbCr 4:2:0 video, a CTU consists of one luma CTB and two chroma CTBs each a quarter of the size of the luma CTB. The maximum CTU size (defined by the largest CTB of the CTU) is 128×128 samples. A CTU can be recursively divided into Coding Units (CUs) according to three partitioning modes: quadtree (division into four equally sized CUs); ternary-tree (division into three CUs of size 1/4th, 2/4th, 1/4th); and binary-tree (division into two equally sized CUs). Additional partitioning can arise in some cases where a CU is split into Transform Units (TUs) of smaller size than the CU size. In intra slices, it is possible to apply separate luma and chroma coding trees, in which case the luma and chroma CTBs can be recursively split according to their own coding trees.

The picture partitioning and the quantization steps used in conventional block-based hybrid coding may cause coding artifacts such as block discontinuities, ringing artifacts, mosquito noise, or texture and edge smoothing. In-loop filtering processes are applied in the encoding and decoding loops to reduce these artifacts. In VVC, four different in-loop filters are specified: Deblocking Filter (DBF) for reducing the blocking artifacts, Sample Adaptive Offset (SAO) for attenuating the ringing artifacts and correcting the local average intensity changes, Adaptive Loop Filtering (ALF) and Cross-Component Adaptive Loop Filtering (CC-ALF) for further correcting the signal based on linear filtering and adaptive clipping. In addition, a specific filtering step called Luma Mapping with Chroma Scaling (LMCS) is also defined. LMCS does not specifically address the coding artifacts reduction but

Manuscript received August 21, 2020; revised March 2, 2021; accepted March 25, 2021. Date of publication April 9, 2021; date of current version October 4, 2021. This article was recommended by Associate Editor T. Wiegand. *(Corresponding author: Peng Yin.)*

Marta Karczewicz and Nan Hu are with Qualcomm Inc., San Diego, CA 92121 USA.

Jonathan Taquet was with Canon, 35510 Cesson-Sevigne, France. He is now with Xiaomi, Beijing 100085, China.

Ching-Yeh Chen is with MediaTek, Hsinchu 30078, Taiwan.

Kiran Misra is with Sharp Laboratories of America, Vancouver, WA 98683 USA.

Kenneth Andersson is with Ericsson, 164 83 Stockholm, Sweden.

Peng Yin and Taoran Lu are with Dolby Laboratories, Inc., Sunnyvale, CA 94085 USA (e-mail: pyin@dolby.com).

Edouard François is with InterDigital, 35510 Cesson Sevigne, France.

Jie Chen is with Alibaba, Beijing 100102, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2021.3072297.

Digital Object Identifier 10.1109/TCSVT.2021.3072297

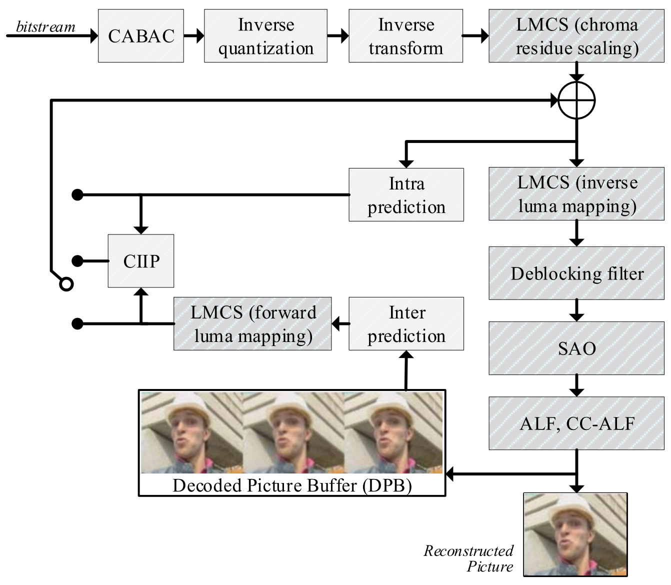


Fig. 1. VVC decoder simplified block diagram; shaded rectangles correspond to in-loop filtering coding blocks.

aims at better using the signal range for improved coding efficiency.

The filters are applied after the picture reconstruction and before saving the picture in the DPB, in the following decoding process order (see Fig. 1): inverse LMCS (luma mapping part), DBF, SAO, ALF, and CC-ALF. The forward LMCS (luma mapping part) is also applied to the luma prediction signal in the inter prediction case and chroma scaling part of LMCS is applied to the chroma residues after inverse transform. The specific order in which these tools are applied is justified by their incremental and complementary benefits. The DBF and SAO are two filters aiming at reducing the artifacts caused by the coding process. The DBF focuses on visual artifacts at block boundaries while SAO complementarily reduces artifacts resulting from the transform coefficients quantization which can arise inside the blocks. These two filters follow inverse LMCS. Applying inverse LMCS right after the sample reconstruction and prior to DBF was motivated by the fact that DBF was designed to perform in the original sample domain, based on subjective criteria, not in the mapped sample domain. ALF and CC-ALF perform a final corrective step that typically targets improving the signal fidelity and thus are placed at the last in-loop filtering stage.

Compared to HEVC, the new in-loop filtering technologies in VVC are ALF, CC-ALF and LMCS. The DBF is conceptually like the HEVC DBF with several enhancements. SAO is identical to that in HEVC. It consists of classifying samples of one CTU into different groups and applying an offset to samples of a group to reduce sample distortions, the group index and offset value being signalled in the bitstream.

SAO, ALF, and CC-ALF add corrective offsets to the signal, while DBF applies a filtering across the block frontiers. Though SAO and ALF/CC-ALF are partly overlapping, they operate in different ways and address different signal artifacts. In addition, SAO is of much lower complexity than ALF and CC-ALF, which makes SAO relevant for low-complexity or low-latency encoders.

ALF is based on adaptive filters, which are typically applied to reduce the mean square error (MSE) between the original and the reconstructed samples using Wiener-based filtering [4]. ALF includes a classification of non-overlapping 4×4 blocks based on their local sample gradients. For each class a specific filter is applied among the different filters signalled in the bitstream. Based on this classification, geometric transformation, such as 90-degree rotation, diagonal or vertical flip, of coefficients within filter shape can also be applied. ALF applies to luma and chroma samples. Further details are provided in Section II.

CC-ALF exploits the correlation between the luma and chroma samples and applies only to the chroma samples. CC-ALF generates a correction of chroma samples using a linearly filtered version of the luma samples located around the same relative location as the chroma samples. A Wiener-filter approach can be used at encoder side to derive the filters for the purpose of MSE reduction. CC-ALF operates in parallel with ALF. CC-ALF details are provided in Section III.

VVC DBF uses the HEVC DBF design, with some adaptations mainly related to the addition of long-tap filters for both luma and chroma and for some specific coding conditions, leading for example to stronger smoothing in case of large coding blocks. VVC deblocking control also benefits from more flexibility and supports a new control mode based on average local luma samples level. More details on DBF are provided in Section IV.

LMCS contains two components: luma mapping (LM) and luma-dependent chroma residue scaling (CS). The basic idea of luma mapping is to make better use of the range of luma code values at a specified bit depth, as some luma code values may not be used in the input video. The CS is designed to compensate for the LM impact on the bit cost repartition between the luma signal and the chroma signal [5]. LMCS details are provided in Section V.

ALF coefficients and LMCS parameters are carried in ALF adaptation parameter sets (ALF APS) and LMCS APSs, respectively [6]. Hence, when multiple slices in the same or different pictures use the same ALF coefficients or the same LMCS parameters, redundant transmission of ALF coefficients and LMCS parameters can be avoided. When ALF or LMCS is applied to a picture or a slice, only IDs of the referenced APSs are signalled in the picture or slice header.

The rest of the paper is organized as follows. Sections II and III describe ALF and CC-ALF, respectively. The deblocking filter is explained in Section IV. Section V addresses the LMCS process. Note that the authors have preferred for clarity purpose presenting the tools in a different order than their application order in the encoding and decoding process. Experimental results are presented and discussed in Section VI whereas Section VII concludes the paper.

II. Adaptive Loop Filter

This section describes ALF techniques employed in VVC. The ALF design including filter shapes, precision and adaptive clipping is presented in Section II.A. Filtering adaptations at sub-block and CTB levels are described in Section II.B

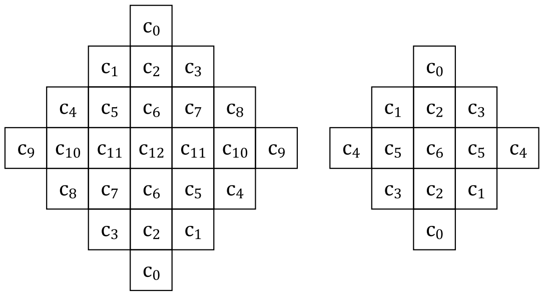


Fig. 2. ALF filter shapes.

and Section II.C, respectively. Syntax design is presented in Section II.D. Reduction of ALF line buffer requirements is discussed in Section II.E. Finally, the ALF encoder design in the VVC test model (VTM) [67] is presented in Section II.F.

A. Filter Shapes, Linear Filtering and Adaptive Clipping

ALF is applied to the output samples of SAO. Two filter shapes, 7 × 7 diamond shape and 5 × 5 diamond shape are supported for luma and chroma components respectively [9], as shown in Fig. 2. In Fig. 2, each square corresponds to a luma or a chroma sample and the center square corresponds to a current to-be-filtered sample. To reduce the signalling overhead and the number of multiplications, the filter coefficients use point-symmetry [4]. Each integer filter coefficient c_i is represented with 7-bit fractional precision [10]. In addition, to preserve DC neutrality, the sum of coefficients of one filter must be equal to 128, which is the fixed-point representation of 1.0 with 7-bit fractional precision:

$$2\sum_{i=0}^{N-2} c_i + c_{N-1} = 128. \tag{1}$$

In Eq. (1), the number of coefficients N is equal to 13 and 7 for 7 × 7 and 5 × 5 filter shape, respectively.

A filtered sample value $\tilde{R}(x, y)$ at coordinates (x, y) is derived by applying coefficient c_i to the reconstructed sample values $R(x, y)$ as follows:

$$\tilde{R}(x,y) = \left[\sum_{i=0}^{N-2} c_i \left(R(x+x_i, y+y_i) + R(x-x_i, y-y_i)\right) + c_{N-1} R(x,y) + 64\right] >> 7, \tag{2}$$

where $(x + x_i, y + y_i)$ and $(x - x_i, y - y_i)$ are the coordinates of the reconstructed samples corresponding to i-th coefficient c_i. Due to the constraint in Eq. (1), Eq. (2) can be written as

$$\begin{aligned}\tilde{R}(x,y) = R(x,y) &+ \left\{\left[\sum_{i=0}^{N-2} c_i \left(R(x+x_i, y+y_i) - R(x,y)\right)\right.\right. \\ &+ \left.\sum_{i=0}^{N-2} c_i \left(R(x-x_i, y-y_i) - R(x,y)\right) + 64\right] \\ &\left. >> 7\right\}.\end{aligned} \tag{3}$$

Based on Eq. (3), the filtered sample $\tilde{R}(x, y)$ is obtained by adding to the reconstructed sample $R(x, y)$ a weighted sum of the differences between the to-be-filtered sample $R(x, y)$ and its neighboring samples.

The coefficients in Eq. (3) are the same for all the samples in the same relative geometric position to the to-be-filtered sample. Unlike linear filters which take into consideration only the geometric closeness of the picture samples, non-linear filters such as bilateral filter [11] can also adjust their coefficients based on the similarity of the sample values. Hence bilateral filter can effectively remove the noise while preserving edges. To allow ALF filter to take into consideration both spatial relationship and value similarity between the samples, the possibility to clip the difference between the neighboring sample value and the current to-be-filtered sample is added [12] to Eq. (3). When non-linear ALF is enabled Eq. (3) is modified as follows:

$$\tilde{R}(x,y) = R(x,y) + \left[\left(\sum_{i=0}^{N-2} c_i f_i + 64\right) >> 7\right], \tag{4}$$

where

$$\begin{aligned} f_i = &\min\left(b_i, \max\left(-b_i, R(x+x_i, y+y_i) - R(x,y)\right)\right) \\ &+ \min\left(b_i, \max\left(-b_i, R(x-x_i, y-y_i) - R(x,y)\right)\right).\end{aligned} \tag{5}$$

b_i is the clipping parameter for a coefficient c_i determined by a clipping index d_i. b_i is derived as follows:

$$b_i = \begin{cases} 2^{BD}, & \text{when } d_i = 0 \\ 2^{BD-1-2d_i}, & \text{otherwise} \end{cases} \tag{6}$$

where BD is the sample bit depth and d_i can be 0, 1, 2 or 3.

B. Luma Sub-Block Level Filter Adaptation

Sub-block level filter adaptation is only applied to luma component. Each 4 × 4 luma block is classified based on its directionality and 2D Laplacian activity [13]. First, the values of sample gradients for horizontal, vertical and two diagonal directions are calculated:

$$\begin{aligned} H_{k,l} &= |2R(k,l) - R(k-1,l) - R(k+1,l)|, \\ V_{k,l} &= |2R(k,l) - R(k,l-1) - R(k,l+1)|, \\ D0_{k,l} &= |2R(k,l) - R(k-1,l-1) - R(k+1,l+1)|, \\ D1_{k,l} &= |2R(k,l) - R(k-1,l+1) - R(k+1,l-1)|. \end{aligned} \tag{7}$$

Based on the sample gradients, sub-block horizontal gradient, g_h, vertical gradient, g_v, and two diagonal gradients, g_{d0} and g_{d1}, are calculated as

$$\begin{aligned} g_h &= \sum_{k=i-2}^{i+5} \sum_{l=j-2}^{j+5} H_{k,l}, \\ g_v &= \sum_{k=i-2}^{i+5} \sum_{l=j-2}^{j+5} V_{k,l}, \\ g_{d0} &= \sum_{k=i-2}^{i+5} \sum_{l=j-2}^{j+5} D0_{k,l}, \\ g_{d1} &= \sum_{k=i-2}^{i+5} \sum_{l=j-2}^{j+5} D1_{k,l}. \end{aligned} \tag{8}$$

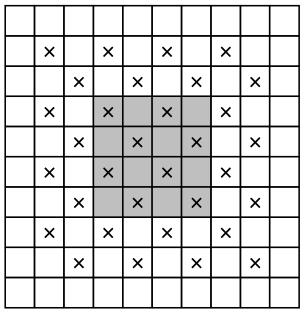


Fig. 3. Subsampled sample gradients for a 4×4 sub-block ALF classification. Gradient values of samples marked with $\times$ are calculated. Gradient values of other samples are set to 0.

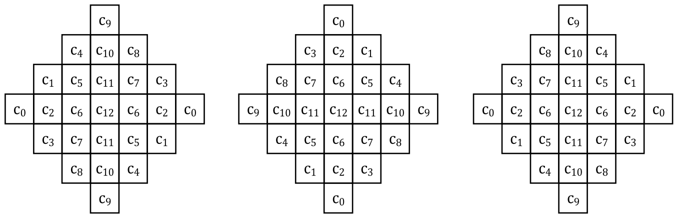


Fig. 4. Geometric transformation of 7×7 diamond filter shape. From left to right: diagonal flip, vertical flip and 90-degree rotation.

Indices i and j refer to the coordinates of the upper left sample in the 4×4 luma block. As it can be seen from Eq. (8), the sum of sample gradients within a 10×10 luma window that covers the target 4×4 block is used for classifying that block. To reduce the complexity, only gradient of every second sample in a 10×10 window is calculated [14] as illustrated in Fig. 3. The values of other sample gradients are set to 0.

Second, to assign the directionality D, the ratio of the maximum and the minimum of the sub-block horizontal and vertical gradients

$$g_{h,v}^{max} = max\left(g_h, g_v\right), g_{h,v}^{min} = min\left(g_h, g_v\right), \quad (9)$$

and the ratio of the maximum and the minimum of two sub-block diagonal gradients

$$g_{d0,d1}^{max} = max\left(g_{d0}, g_{d1}\right), g_{d0,d1}^{min} = min\left(g_{d0}, g_{d1}\right), \quad (10)$$

are compared against each other and with a set of thresholds t_1 and t_2:

Step 1: If both $g_{h,v}^{max} \le t_1 \cdot g_{h,v}^{min}$ and $g_{d0,d1}^{max} \le t_1 \cdot g_{d0,d1}^{min}$, D is set to 0 (block is categorized as "texture").

Step 2: If $g_{h,v}^{max} \big/ g_{h,v}^{min} > g_{d0,d1}^{max} \big/ g_{d0,d1}^{min}$, the directionality D is calculated in Step 3, otherwise in Step 4.

Step 3: If $g_{h,v}^{max} > t_2 \cdot g_{h,v}^{min}$, D is set to 2 (block is categorized as "strong horizontal / vertical"), otherwise D is set to 1 (block is categorized as "weak horizontal / vertical").

Step 4: If $g_{d0,d1}^{max} > t_2 \cdot g_{d0,d1}^{min}$, D is set to 4 (block is categorized as "strong diagonal"), otherwise D is set to 3 (block is categorized as "weak diagonal").

Each subsequent step in the above calculation of D is only executed if there is no value assigned to D in the previous

TABLE I

GEOMETRIC TRANSFORMATION BASED ON SUB-BLOCK GRADIENT VALUES

Sub-block gradient values	Transformation
$g_{d1} < g_{d0}$ and $g_h < g_v$	No transformation
$g_{d1} < g_{d0}$ and $g_v \le g_h$	Diagonal flip
$g_{d0} \le g_{d1}$ and $g_h < g_v$	Vertical flip
$g_{d0} \le g_{d1}$ and $g_v \le g_h$	90-degree rotation

steps. Third, an activity value A is calculated as

$$A = \left(\sum_{k=i-2}^{i+5} \sum_{l=j-2}^{j+5} \left(V_{k,l} + H_{k,l}\right)\right) >> (BD - 2)\,. \quad (11)$$

A is further mapped to the range of 0 to 4: $\hat{A} = Q_{min(A,15)}$ where $\{Q_n\} = \{0, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4\}$. Finally, each 4×4 luma block is categorized into one of the 25 classes:

$$C = 5D + \hat{A}. \quad (12)$$

Each class can have its own filter assigned.

Before filtering each 4×4 luma block, a geometric transformation, such as 90-degree rotation, diagonal or vertical flip, is applied to the filter coefficients, as illustrated in Fig. 4, depending on the sub-block gradient value as specified in Table I. This is equivalent to applying these transformations to the samples in the filter support region. The goal is to align the directionality of the different blocks, in order to reduce the number of ALF classes and, in turn, filter coefficients. Applying the geometric transformation allows a 4×4 block with a horizontal edge and a 4×4 block with a vertical edge to both have the same directionality D.

C. Coding Tree Block Level Filter Adaptation

In addition to the luma 4×4 block-level filter adaptation, ALF supports CTB-level filter adaptation [15], [16]. A luma CTB can use a filter set calculated for the current slice or one of the filter sets calculated for the already coded slices. It can also use one of the 16 offline trained filter sets. Within each luma CTB, which filter from the chosen filter set should be applied to each 4×4 block, is determined by the class C calculated in Eq. (12) for this block.

Chroma uses only CTB-level filter adaptation. Up to 8 filters can be used for chroma components in a slice. Each CTB can select one of these filters.

D. Syntax Design

Filter coefficients and clipping indices are carried in ALF APSs. An ALF APS can include up to 8 chroma filters and one luma filter set with up to 25 filters. An index i_C is also included for each of the 25 luma classes. Classes having the same index i_C share the same filter. By merging different classes, the number of bits required to represent the filter coefficients is reduced. The absolute value of a filter coefficient is represented using a 0th order Exp-Golomb code followed by a sign bit for

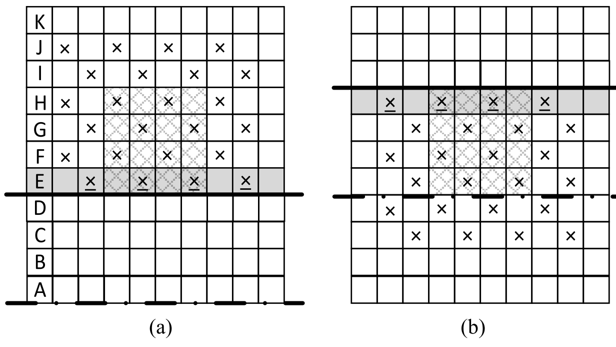


Fig. 5. ALF 4×4 luma sub-block classification at line buffer boundaries. Dashed lines are horizontal CTU boundaries and solid bold lines are line buffer boundaries. 4×4 luma blocks to be classified are shaded. Gradient values of samples marked with $\times$ and $\underline{\times}$ are calculated. Gradient values of other samples are set to 0. (a): Current 4×4 block is above a line buffer boundary. (b): Current 4×4 block is below a line buffer boundary.

a non-zero coefficient. When clipping is enabled, a clipping index is also signalled for each filter coefficient using a two-bit fixed-length code. The storage needed for ALF coefficients and clipping indices within an APS is at most 3480 bits. Up to 8 ALF APSs can be used by the decoder at the same time.

Filter control syntax elements include two types of information. First, ALF on/off flags are signalled at sequence, picture, slice and CTB levels. Chroma ALF can be enabled at picture and slice level only if luma ALF is enabled at the corresponding level. Second, filter usage information is signalled at picture, slice and CTB level, if ALF is enabled at that level. Referenced ALF APSs IDs are coded at a slice level or at a picture level if all the slices within the picture use the same APSs. Luma component can reference up to 7 ALF APSs and chroma components can reference 1 ALF APS. For a luma CTB, an index is signalled indicating which ALF APS or offline trained luma filter set is used. For a chroma CTB, the index indicates which filter in the referenced APS is used.

E. Line Buffer Reduction

To reduce the storage requirement for ALF, VVC employs line buffer boundary processing. In VVC, line buffer boundaries are placed 4 luma samples and 2 chroma samples above horizontal CTU boundaries. When applying ALF to a sample on one side of a line buffer boundary, samples on the other side of the line buffer boundary cannot be used.

Fig. 5 gives two examples in which a 4×4 luma block is adjacent to a line buffer boundary. Without the line buffer boundary processing when applying ALF to the 4×4 luma block in rows E to H in Fig. 5(a), samples from rows B to K, filtered using DBF and SAO, are required. DBF and SAO filters cannot be applied to rows A to D until the lower CTU is available. Hence, without the line buffer boundary processing ALF could not be applied to samples in rows E to H until the lower CTU is available. As a result, 7 luma rows from E to K, in addition to rows A to D, would have to be stored in the line buffer for luma ALF. Similarly, 4 additional chroma rows would have to be stored in the line buffer for chroma ALF [17].

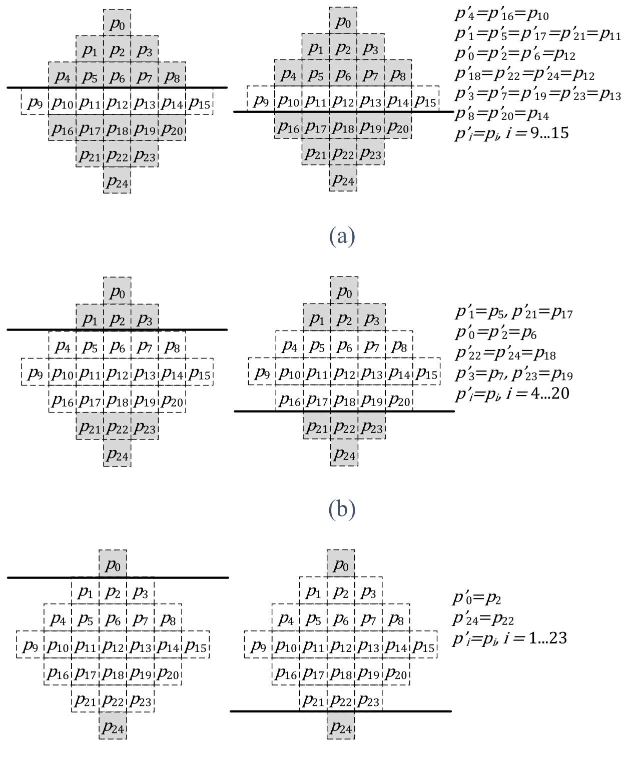


Fig. 6. Symmetrical sample padding of luma ALF filtering when the filter shape of a to-be-filtered sample crosses a line buffer boundary. Bold lines are line buffer boundaries. Shaded samples represent padded samples.

Fig. 5 also illustrates the calculation of sample gradient values, when classifying a 4×4 luma block adjacent to the line buffer boundary. To calculate the sample gradient values adjacent to the line buffer boundary, which are marked as $\underline{\times}$, repetitive padding is applied to replace samples which cannot be used. For example, in Fig. 5(a), samples in row D are replaced with samples in row E. All sample gradient values on the other side of the line buffer boundary are set as 0. Since we set some of the sample gradient values to 0, reducing the sum of the sample gradients, the activity derivation in Eq. (11) is scaled as follows:

$$A = \left(\sum_{k=i-2}^{i+5} \sum_{l=j-2}^{j+5} \left(V_{k,l} + H_{k,l} \right) \cdot 3 \right) >> (BD - 1)\,. \tag{13}$$

The line buffer boundary filtering applies symmetrical sample padding illustrated in Fig. 6, where p_{12} marks the to-be-filtered sample, p_0 to p_{24} are the sample values after SAO and p'_0 to p'_{24} are the modified sample values. When the filter shape of the to-be-filtered sample does not cross the line buffer boundary, sample values after SAO are used in the filtering process. Otherwise, the modified values are used in the filtering process. Compared to the repetitive padding, symmetrical sample padding has been demonstrated to create less noticeable visual artifacts. However, when a sample to be filtered is located in a row closest to the line buffer boundary, as shown in Fig. 6(a), the 2D filter is equivalent to a horizontal filter, which still can introduce noticeable visual artifacts.

These artifacts are minimized by reducing the filter strength by a factor of 8 [18], leading to the following formula, where the shift by 7 in Eq. (4) is replaced by the shift by 10:

$$\tilde{R}(x,y) = R(x,y) + \left[\left(\sum_{i=0}^{N-2} c_i f_i + 512\right) >> 10\right] \quad (14)$$

F. ALF Encoder Design in VTM

The ALF encoder implementation in VTM-9.0 [67] is described here. The encoder calculates values of ALF syntax elements by minimizing a rate-distortion cost [19], which is a weighted sum of the distortion, measured as the square error between the original samples and the samples after applying the ALF filter, and the number of bits required to transmit ALF syntax elements. Filter coefficients are calculated by solving Wiener-Hopf equations [4]. A square error estimation method proposed in [4] is adopted allowing to calculate the filtering distortion without performing actual filtering operations. The statistics required to calculate filter coefficients and estimate distortion are collected for all possible combinations of clipping indices. The statistics are collected separately for each CTB, and in case of luma component, for each class of each CTB.

1) Luma Component: For each picture, a new filter set for luma component based on this picture's statistics, denoted as $F_{Y,D}$, is obtained as follows:

1) The encoder derives the filter set $F_{Y,D}$, by merging statistics of the CTBs for which ALF is enabled. In the first iteration, it is assumed that ALF is enabled for all CTBs.
2) Whether to apply ALF filter is determined for each CTB based on the rate-distortion cost calculated using the derived filters and the statistics of the CTB.

Steps 1) and 2) are repeated 4 times.

When designing a luma filter set in Step 1), the encoder first calculates a filter for each of the 25 luma classes. Then a merging algorithm is applied to these 25 filters. In each iteration, by merging two filters, the algorithm reduces the number of filters by 1. To determine which two filters should be merged, for every pair of the remaining filters, the encoder redesigns a filter by merging two filters and the corresponding statistics, respectively. Using the redesigned filter, the distortion is then estimated. The encoder merges the pair with the smallest distortion. 25 filter sets are obtained, the first set having 25 filters, the second one 24 filters, and so on until the 25th one contains a single filter. The set which minimizes rate-distortion cost, including bits required to code filter coefficients, is selected.

When designing a filter, the clipping indices and the $N-1$ filter coefficients are calculated iteratively until there is no decrease of the square error. In each iteration, the values of the clipping indices are updated one by one, starting with index d_0 and continuing till index d_{N-2} is reached. When updating the index, up to 3 options are tested: keeping its value unchanged, increasing by 1 or decreasing by 1. The filter coefficients and the approximate distortion are calculated for these 3 values and the value which minimizes square error is selected. At the start of the first iteration the values of clipping indices are initialized to 2, or when merging two filters, the value of d_i is set to the average of the corresponding clipping indices for these filters.

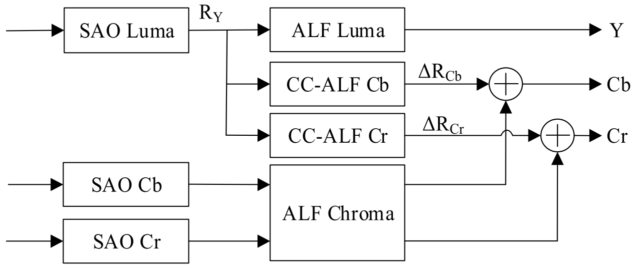


Fig. 7. CC-ALF architecture.

Luma component can also use the offline trained filter sets and the luma filter sets carried in available ALF APSs. By accessing APSs, from the most recently signalled to the last one, the encoder obtains up to 7 luma filter sets, which are denoted as $F^i_{Y,APS}$, where $i = 0, \ldots, N_{APS} - 1$. The final values of the luma ALF syntax elements are obtained by selecting the best combination of filter sets $\boldsymbol{F}_{ij}$, where $i = 0$, 1 and $j = 0, \ldots, N_{APS}$:

- 16 offline trained filter sets,
- $F_{Y,D}$, if i is 1,
- and $F^0_{Y,APS}, \ldots, F^{j-1}_{Y,APS}$, if $j > 0$.

2) Chroma Components: Based on the current picture statistics, a chroma filter set $F_{C,D}$ is calculated. 8 chroma filter sets are derived, denoted as $F^i_{C,D}$ where $i = 1, 2, \ldots, 7, 8$. Filter set $F^i_{C,D}$ contains i filters. The filters set $F^i_{C,D}$ is obtained as follows:

1) The current picture is uniformly partitioned into i regions. For each region, a chroma filter is calculated by merging the statistics of all CTBs in this region.
2) For each chroma CTB in the picture, ALF on/off flag and the filter index from set $F^i_{C,D}$ is determined as to minimize the estimated rate-distortion cost.
3) Each filter in $F^i_{C,D}$ is re-designed, by merging statistics of chroma CTBs for which the current to-be-redesigned filter was selected in step 2).
4) Steps 2) and 3) are repeated $i + 1$ times.

The filter set $F^i_{C,D}$ that minimizes rate-distortion cost is selected as chroma filter set $F_{C,D}$. Finally, the encoder selects $F_{C,D}$ or one of the filter sets from the available APSs to be used for the chroma components.

III. Cross-Component Adaptive Loop Filter

CC-ALF uses the luma sample values to refine the chroma sample values within the ALF process. As shown in Fig. 7, a linear filtering operation takes the luma sample values as input and generates the correction values for the chroma sample values. The correction is generated independently for each chroma component i, $i \in \{Cb, Cr\}$ and can be represented by:

$$\Delta R_i(x,y) = \sum_{(x_0,y_0)\in S_i} R_Y\left(x_C + x_0, y_C + y_0\right) c_i(x_0, y_0), \quad (15)$$

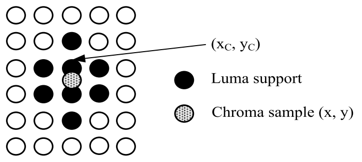


Fig. 8. Relative location of filtered chroma sample and its support in the luma plane for 4:2:0 chroma format with chroma location type 0.

where (x, y) is the sample location of the chroma component i, (x_C, y_C) is the luma sample location derived from (x, y), (x_0, y_0) are the filter support offset around (x_C, y_C), S_i is the filter support region in luma for the chroma component i, and $c_i\,(x_0, y_0)$ represent the filter coefficients of the component i. The luma location (x_C, y_C) is determined based on the spatial scaling factor between the luma and chroma planes. The sample values in the luma support region are also inputs to the ALF luma stage and correspond to the output of the SAO stage. Applying a cross-component filtering operation in the ALF process was first proposed in [20] and subsequently refined in [24]–[32]. While this work is primarily focused on the VVC ALF process, in the past other work has also considered the correlation between the luma and the chroma channels, for example [22], [23], [34]. Some of the design decisions for CC-ALF and their rationale were previously described in [21]. The CC-ALF design has been further refined with the aim of reducing complexity, and the final VVC design is described in subsequent sections.

A. Filter Shape and Precision

As shown in Fig. 8, the CC-ALF filter has a diamond shape. Compared to a rectangular shape, the diamond shape reduces the number of coefficients and, consequently, the number of multiply-accumulate (MAC) operations for the filter. As seen in Fig. 8, for a 4:2:0 video sequence, with chroma location type 0, i.e., when the chroma samples are horizontally co-sited with the even numbered columns of the luma samples and vertically interstitial between the rows of the luma samples, the center of the diamond is aligned with a chroma sample location. To further reduce MAC operations, the size of the diamond was reduced from the original design of 5×6 to 3×4 [24]–[26]. Overall, when considering all color components, for 4:2:0 chroma format, the number of MACs increases from 15 per luma sample for ALF-only to 19 for ALF and CC-ALF, which represents an increase of about 25%.

CC-ALF coefficients have a greater degree of flexibility compared to regular ALF coefficients, since no symmetry constraints are enforced. This flexibility is desirable because the relative position of the luma and the chroma samples can vary based on the chroma location type and the chroma format. However, two limitations are enforced:

1) To preserve DC neutrality, the sum of CC-ALF coefficient values is required to be zero [28]. As a result, only seven of the eight CC-ALF coefficients need to be signalled in the bitstream, and the coefficient at location (x_C, y_C) is derived at the decoder.
2) The absolute value of CC-ALF coefficients is restricted to be either zero or an integer power of two, specifically $\{0, 1, 2, 4, 8, 16, 32, 64\}$. This enables implementations to use variable bit-shift operations in place of multiplications for CC-ALF [27], if desired.

Since the absolute value of a coefficient can be indicated using three bits, in the worst-case, the storage needed for CC-ALF filter coefficients within an APS is 224 bits.

B. Latency and Buffering

As has been described in [21], CC-ALF can be executed concurrently with ALF filters using the data flow shown in Fig. 7. The correction values output by CC-ALF are also clipped to $\left[-2^{BitDepth-1}, 2^{Bitdepth-1} - 1\right]$ to reduce storage requirements. For example, when CC-ALF is first applied concurrently with luma ALF, and chroma ALF is applied later, the memory access pattern for CC-ALF input is the same as that for luma ALF.

As described in Section II.E, the luma and the chroma line buffer boundaries are four and two samples, respectively, above the CTU boundary. For the 4:2:0 chroma format, this results in line buffer boundaries that are aligned for chroma and luma. However, for 4:2:2 and 4:4:4 chroma formats, the chroma and the luma line buffer boundaries are not aligned with each other. As a result of this misalignment, for 4:2:2 and 4:4:4 chroma formats, CC-ALF is not applied to the rows three and four samples above the CTU boundary [31].

Tools that use luma samples to predict chroma samples, such as Cross Component Linear Model (CCLM) intra prediction, may introduce latency as luma samples need to be fully processed before the reconstruction of chroma samples can begin. However, due to its design, no such latency issue exists for the CC-ALF tool.

C. Syntax Design

To mitigate the impact of the filter size reduction on coding efficiency, the maximum number of filters per chroma component of a picture was increased from one in [20] to four [26] in the final design of VVC. A different set of CC-ALF coefficients can be selected for each CTU of a chroma component. As is the case for the regular ALF coefficients, CC-ALF coefficients are signalled within an ALF APS. Each ALF APS contains up to four CC-ALF filters for each chroma component. While CC-ALF can be enabled at a sequence level, it can only be enabled if ALF is also enabled for the sequence. Similarly, CC-ALF can be enabled at picture and slice level only if luma ALF is enabled at the corresponding level [29].

IV. Deblocking Filter

In a block-based video codec such as VVC, discontinuities can appear at the transform and prediction block boundaries. These discontinuities can lead to visual artifacts that are referred to as blocking artifacts. The main reason for blocking artifacts is the difference in mean sample values between adjacent blocks. A deblocking filter aims to achieve a smooth transition across the block boundaries while avoiding removal of natural edges. What is regarded as natural edges depends

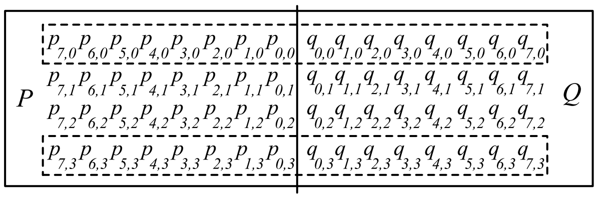

P
Q
$p_{7,0}\,p_{6,0}\,p_{5,0}\,p_{4,0}\,p_{3,0}\,p_{2,0}\,p_{1,0}\,p_{0,0}$ $q_{0,0}\,q_{1,0}\,q_{2,0}\,q_{3,0}\,q_{4,0}\,q_{5,0}\,q_{6,0}\,q_{7,0}$
$p_{7,1}\,p_{6,1}\,p_{5,1}\,p_{4,1}\,p_{3,1}\,p_{2,1}\,p_{1,1}\,p_{0,1}$ $q_{0,1}\,q_{1,1}\,q_{2,1}\,q_{3,1}\,q_{4,1}\,q_{5,1}\,q_{6,1}\,q_{7,1}$
$p_{7,2}\,p_{6,2}\,p_{5,2}\,p_{4,2}\,p_{3,2}\,p_{2,2}\,p_{1,2}\,p_{0,2}$ $q_{0,2}\,q_{1,2}\,q_{2,2}\,q_{3,2}\,q_{4,2}\,q_{5,2}\,q_{6,2}\,q_{7,2}$
$p_{7,3}\,p_{6,3}\,p_{5,3}\,p_{4,3}\,p_{3,3}\,p_{2,3}\,p_{1,3}\,p_{0,3}$ $q_{0,3}\,q_{1,3}\,q_{2,3}\,q_{3,3}\,q_{4,3}\,q_{5,3}\,q_{6,3}\,q_{7,3}$

Fig. 9. Four samples long vertical boundary segment formed by block P and block Q. Deblocking decisions are based on line 0 and line 3.

on the QP (quantization parameter) used for compressing transform coefficients. Lower QP corresponds to presence of natural edges of smaller magnitude, and vice-versa.

The VVC DBF is based on the HEVC design [35]. In HEVC, the first step is to determine the boundary strength (bS) on an 8×8 sample grid. The boundary strength can take one of the three possible values: 0, 1, and 2. For the luma component, only block boundaries with bS values equal to one or two are filtered. For chroma, only boundaries with bS equal to 2 are filtered. bS is determined for a four samples long boundary segment between block P and Q, as shown in Fig. 9, according to the following rules:

- If P or Q is intra coded, bS is set to 2.
- Otherwise, if there is a significant difference in motion between P and Q or if P or Q have non-zero transform coefficients, bS is set to 1.
- Otherwise, bS is set to 0.

The second step is making a filtering decision based on the spatial activity analysis of lines 0 and 3 of the boundary segment. When the spatial activity is below thresholds derived from QP-dependent parameters t_c and β, the DBF is applied for all four lines of the boundary segment. Otherwise, no deblocking process is applied. Both t_c and β increases with QP. This avoids applying the DBF in the presence of natural edges. The amount of smoothing of the DBF is further controlled by t_c. Clipping is used to make sure that a filtered value does not deviate more than t_c from the sample value before filtering. The QP used for determining t_c is increased by 2 if bS is equal to 2. This enables larger modifications for intra coded blocks which typically have blocking artifacts with greater strength. The filtering decisions and operations are applied first to vertical block boundaries and then to horizontal block boundaries.

VVC allows larger block sizes than HEVC does, for example, blocks of 128×128 samples. When the HEVC DBF is applied as is, visible artifacts can still exist, especially in relatively smooth areas. The DBF in VVC extends the HEVC DBF design to address these artifacts. This section gives an overview of the VVC deblocking design. For further details the reader is referred to the VVC specification [1] and to the corresponding JVET input contributions.

A. Luma Deblocking

Due to the flexible block sizes and new coding tools in VVC, the luma deblocking is applied on a 4×4 sample grid [43] for boundaries between CUs and TUs [38], [39] and on an 8×8 grid for boundaries between PUs [37] inside CUs, as shown in Fig. 10. In this section, "PUs" refer to prediction sub-blocks within a CU that use affine or sub-block based temporal motion vector prediction (SbTMVP) coding mode. In the following sub-sections filtering decisions and filtering operations for luma are described.

a) Filtering decisions: Before deriving bS and performing filtering decisions based on spatial activity, it is required to decide whether to use long-tap filters and determine appropriate filter lengths. This step is carried out to ensure that no spatial dependency exists between deblocking of the adjacent vertical or horizontal block boundaries. The deblocking length is defined as the number of samples to be filtered for each line for adjacent blocks P and Q (see Fig. 9). The deblocking length of P and Q is denoted as S_P and S_Q respectively. The number of samples used for filtering decisions and filtering operations are $(S_P + 1)$ and $(S_Q + 1)$. The value of S_P and S_Q depends on the size of the block side orthogonal to the block boundary for block P and Q, respectively. S_P/S_Q of the CU/TU boundary is set initially as follows:

- If the CU/TU block side size is greater than or equal to 32, S_P/S_Q is set to 7.
- Otherwise, if the CU/TU block side size is less than or equal to 4, S_P and S_Q are set to 1.
- Otherwise, remaining uninitialized S_P/S_Q is set to 3.

If a CU uses PUs, S_P/S_Q of the CU/TU boundary is set as follows:

- If CU/TU boundary is 8 samples distant from a PU boundary, corresponding S_P or S_Q is restricted to be less than or equal to 5.

S_P/S_Q of the PU boundary is calculated as:

- If the PU boundary is 8 samples distant from a CU/TU boundary, S_P and S_Q are restricted to be less than or equal to 2.
- Otherwise, if the PU boundary is 4 samples distant from a CU/TU boundary, S_P and S_Q are set to 1.
- Otherwise, S_P and S_Q are set to 3.

S_P on the upper side of a horizontal CTU boundary, is restricted to be less than or equal to 3. The deblocking lengths, $S_P + S_Q$, can thus be $7+7$, $7+5$, $5+7$, $5+5$, $7+3$, $3+7$, $5+3$, $3+5$, $3+3$ or $1+1$ for CU/TU boundaries and $3+3$, $2+2$ or $1+1$ for PU boundaries inside a CU. These deblocking lengths can be reduced further in the subsequent steps of filtering decisions.

After determining the deblocking filter lengths, bS is derived. Compared to HEVC, bS derivation is modified to accommodate new VVC coding tools. For example, if both blocks adjacent to the boundary use block-level differential pulse code modulation (BDPCM), bS is set to 0, to avoid smoothing such samples. If one of the blocks uses CIIP, bS is set to 2, because that mode is based on intra prediction and stronger filtering should be applied. If both blocks have different prediction modes (inter, palette, intra-picture block copy), bS is set to 1. If both blocks use intra-picture block copy (IBC), bS is set to 1 when their respective IBC vectors are different. bS derivation based on differences in motion in adjacent blocks is only derived for CU/PU boundaries to avoid filtering TU boundaries based on motion. Half-sample

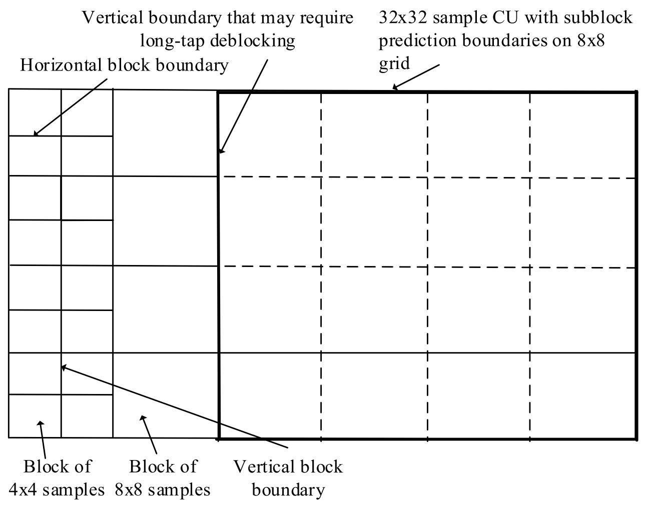

Vertical boundary that may require long-tap deblocking
32x32 sample CU with subblock prediction boundaries on 8x8 grid
Horizontal block boundary
Block of 4x4 samples
Block of 8x8 samples
Vertical block boundary

Fig. 10. Illustration of vertical and horizontal block boundaries on the 4×4 grid, 32×32 samples CU with PUs on 8×8 grid and vertical boundary that may require long-tap deblocking.

differences in motion can set *bS* to 1 because of the high precision of motion in VVC.

In the next step, decisions based on spatial activity are made for the cases of non-zero *bS*. If the deblocking length is larger than 3 on at least one side, the spatial activity decision for the long-tap deblocking filter applies [40], [41]. Otherwise, the decision process for short-tap deblocking filter applies. The short-tap decision process is identical to that of HEVC and is based on the following 4 conditions [35]:

$$\begin{aligned} dPQ_0 + dPQ_3 &< \beta, \\ 2dPQ_i &< thr1, \\ (sP_i + sQ_i) &< thr2, \\ |p_{0,i} - q_{0,i}| &< 2.5t_c, \end{aligned} \tag{16}$$

where i is equal to 0 or 3. dPQ_i checks for natural edges and is equal to the sum of $|p_{0,i} - 2p_{1,i} + p_{2,i}|$ and $|q_{0,i} - 2q_{1,i} + q_{2,i}|$. sP_i and sQ_i check the signal flatness and are equal to $|p_{0,i} - p_{3,i}|$ and $|q_{0,i} - q_{3,i}|$, respectively. $thr1$ and $thr2$ are equal to $\beta >> 2$ and $\beta >> 3$, respectively. If the deblocking length is equal to 1 or 2, only the first condition of Eq. (16) is checked. If only the first condition holds, the short-tap normal filter is selected to be used. If all conditions hold, the short-tap strong filter is selected.

The spatial activity decision for the long-tap filter extends Eq. (16) to include more samples. When S_P is greater than 3, dPQ_i and sP_i also depend on $|p_{3,i} - 2p_{4,i} + p_{5,i}|$ and $|p_{3,i} - p_{S_P,i}|$, respectively. If S_P is equal to 7, $|p_{4,i} - p_{5,i} - p_{6,i} + p_{7,i}|$ is additionally used when calculating sP_i. Derivation of dPQ_i and sQ_i is similarly modified when S_Q is greater than 3.

To avoid over smoothing of the long-tap deblocking filter, the thresholds $thr1$ and $thr2$ in Eq. (16) are reduced to be equal to $\beta >> 4$ and $3\beta >> 5$, respectively. If, according to the spatial activity decision, long-tap deblocking is not applied, then the decoder falls back to the short-tap deblocking decision.

b) Filtering operations: If S_P or S_Q is larger than 3 the long-tap deblocking filter is applied. If both S_P and S_Q are equal to 3 the short-tap strong deblocking filter is applied. Otherwise, the short-tap normal deblocking filter is applied. The short-tap deblocking filters are identical to the HEVC deblocking filters [35] with the exception of a modification to the clipping of the short-tap strong deblocking filter. The modification enables position-dependent clipping to control the difference between filtered values and the sample values before filtering. The clipping range is reduced when the distance from the boundary increases. The clipping range is $\pm 3t_c$, $\pm 2t_c$ and $\pm t_c$ instead of $\pm 2t_c$, for all positions. The long-tap deblocking filter also applies position-dependent clipping. If the deblocking length is 7, the ranges are $\pm 6t_c$, $\pm 5t_c$, $\pm 4t_c$, $\pm 3t_c$, $\pm 2t_c$, $\pm t_c$ and $\pm t_c$. Otherwise, if the deblocking length is 5, the ranges are $\pm 6t_c$, $\pm 5t_c$, $\pm 4t_c$, $\pm 3t_c$ and $\pm 2t_c$. Otherwise, the ranges are $\pm 6t_c$, $\pm 4t_c$ and $\pm 2t_c$.

The long-tap deblocking filter is designed to preserve inclined surfaces or linear signals across a block boundary [40]. The long-tap deblocking filter, which is applied before the position-dependent clipping, is described in Eq. (17) and in Table II.

$$\begin{aligned} p'_{k,i} &= (f_k refM_i + (64 - f_k) refP_i + 32) >> 6, \\ q'_{l,i} &= (g_l refM_i + (64 - g_l) refQ_i + 32) >> 6, \\ refP_i &= \left(p_{S_P,i} + p_{S_P+1,i} + 1\right) >> 1, \\ refQ_i &= \left(q_{S_Q,i} + q_{S_Q+1,i} + 1\right) >> 1, \end{aligned} \tag{17}$$

where, $i = 0$ to 3, $k = 0$ to $S_P - 1$, $l = 0$ to $S_Q - 1$, $p'_{k,i}$ is a filtered sample in block P and $q'_{l,i}$ is a filtered sample in block Q. Other variants of the long-tap deblocking filter are obtained by interchanging p with q, S_P with S_Q and $\mathbf{f}$ with $\mathbf{g}$.

B. Chroma Deblocking

The chroma deblocking is applied on an 8×8 sample grid on boundaries of both CUs and TUs. In the following subsections, filtering decisions and filtering operations for chroma are described.

a) Filter decision: As for luma, an additional step to determine the deblocking lengths is performed before the *bS* determination. The deblocking lengths, S_P and S_Q are set to 3, when the CU/TU block side sizes orthogonal to the block boundary are both greater than or equal to 8 in chroma samples. Otherwise, S_P and S_Q are set to 1. The deblocking length for the upper side of a horizontal CTU boundary, S_P, is restricted to be 1 [42]. The deblocking lengths, $S_P + S_Q$, can thus be $3 + 3$, $1 + 3$ or $1 + 1$.

After deriving the deblocking lengths, the boundary strength is determined. Compared to HEVC, the *bS* derivation is modified such that *bS* is set to 1 when at least one side of the chroma component block boundary has transform coefficients. For the new VVC coding modes BDPCM and CIIP, *bS* is derived in the same way as for luma. If both S_P and S_Q are equal to 1 and *bS* is not equal to 2, S_P and S_Q are reduced to 0. When S_Q is equal to 3 and *bS* is non-zero, an additional decision based on spatial activity is made.

The spatial activity decision for the long-tap chroma filter is determined as in Eq. (16). It is used for lines 0 and 1 of

TABLE II
FILTER KERNELS FOR LONG-TAP DEBLOCKING FILTERING

S_P	S_Q	$refM_i$	Filter
7	7	$\left(\sum_{m=1}^{6}(p_{m,i}+q_{m,i})+2\cdot(p_{0,i}+q_{0,i})+8\right)>>4$	f = g = {59,50, 41,32,23,14,5}
5	7	$\left(\sum_{m=0}^{5}(p_{m,i}+q_{m,i})+\sum_{m=0}^{1}(p_{m,i}+q_{m,i})+8\right)>>4$	f = {58,45,32,19, 6}, g = {59,50, 41,32,23,14,5}
3	7	$\left(2\sum_{m=0}^{2}(p_{m,i})+\sum_{m=0}^{6}(q_{m,i})+p_{0,i}+p_{1,i}+q_{0,i}+8\right)>>4$	f = {53,32,11}, g = {59,50, 41,32,23,14,5}
5	5	$\left(\sum_{m=0}^{4}(p_{m,i}+q_{m,i})+\sum_{m=0}^{2}(p_{m,i}+q_{m,i})+8\right)>>4$	f = g = {58,45, 32,19,6}
3	5	$\left(\sum_{m=0}^{3}(p_{m,i}+q_{m,i})+4\right)>>3$	f = {53,32,11}, g = {58,45,32,19, 6}

chroma samples instead of lines 0 and 3 of luma samples when the chroma format is 4:2:0 and otherwise for lines 0 and 3 of chroma samples [47]. The long-tap chroma deblocking filter is applied only when the long-tap chroma spatial activity decision holds, otherwise S_P and S_Q are reduced to 1.

b) Filter operation: If both S_P and S_Q are equal to 3, the long-tap chroma deblocking filter is applied (before clipping the output by $\pm t_c$) for lines $i = 0$ to 3 as follows:

$$\begin{aligned}
p'_{0,i} &= \left(p_{3,i}+p_{2,i}+p_{1,i}+2p_{0,i}+q_{0,i}+q_{1,i}+q_{2,i}+4\right)\\
&\quad >> 3,\\
q'_{0,i} &= \left(p_{2,i}+p_{1,i}+p_{0,i}+2q_{0,i}+q_{1,i}+q_{2,i}+q_{3,i}+4\right)\\
&\quad >> 3,\\
p'_{1,i} &= \left(2p_{3,i}+p_{2,i}+2p_{1,i}+p_{0,i}+q_{0,i}+q_{1,i}+4\right)\\
&\quad >> 3,\\
q'_{1,i} &= \left(p_{1,i}+p_{0,i}+q_{0,i}+2q_{1,i}+q_{2,i}+2q_{3,i}+4\right) >> 3,\\
p'_{2,i} &= \left(3p_{3,i}+2p_{2,i}+p_{1,i}+p_{0,i}+q_{0,i}+4\right) >> 3,\\
q'_{2,i} &= \left(p_{0,i}+q_{0,i}+q_{1,i}+2q_{2,i}+3q_{3,i}+4\right) >> 3, \qquad (18)
\end{aligned}$$

Otherwise, if S_P is equal to 1 and S_Q is equal to 3, the long-tap chroma deblocking filter is applied with reduced support in block P as follows:

$$\begin{aligned}
p'_{0,i} &= \left(3p_{1,i}+2p_{0,i}+q_{0,i}+q_{1,i}+q_{2,i}+4\right) >> 3\\
q'_{0,i} &= \left(2p_{1,i}+p_{0,i}+2q_{0,i}+q_{1,i}+q_{2,i}+q_{3,i}+4\right) >> 3\\
q'_{1,i} &= \left(p_{1,i}+p_{0,i}+q_{0,i}+2q_{1,i}+q_{2,i}+2q_{3,i}+4\right) >> 3\\
q'_{2,i} &= \left(p_{0,i}+q_{0,i}+q_{1,i}+2q_{2,i}+3q_{3,i}+4\right) >> 3 \qquad (19)
\end{aligned}$$

Otherwise, if both S_P and S_Q are equal to 1, the short-tap chroma deblocking filter, identical to the HEVC chroma deblocking filter, is applied.

C. Adaptive Control of Deblocking

Because the range of QPs is increased in VVC compared to HEVC, the QP-dependent look-up tables that determine t_c and β have been extended correspondingly [44], [46]. The t_c table also uses a higher precision [45] since it is defined for 10-bit video instead of for 8-bit video. For 8-bit video the high precision t_c is right-shifted by 2 which produces the same t_c as in HEVC for the corresponding QP.

D. Sequence, Picture, Slice and Luma Level Adaptivity

Because different sequences can have different characteristics, the parameters t_c and β can be adjusted for each slice and/or picture. Compared to HEVC, more flexible deblocking control parameters are offered, by allowing signalling the parameters for each color component [48].

The deblocking can also be controlled by a new feature, luma-adaptive deblocking [36], where the amount of deblocking is controlled by the average local luma level, $lumaLevel$, defined for the boundary segment as follows:

$$lumaLevel = \left(p_{0,0}+p_{0,3}+q_{0,0}+q_{0,3}\right) >> 2 \qquad (20)$$

The luma-adaptive deblocking is locally adapted by modifying the QP, using a luma-level dependent QP offset table signalled at sequence level. This new feature can be useful to control the amount of deblocking differently for content with highly non-linear transfer functions such as Perceptually Quantized (PQ) and Hybrid Log Gamma (HLG) [8]. During the development of VVC this tool was shown to also provide subjective benefit for standard dynamic range (SDR) [49].

E. Computational Complexity and Parallelism

Although deblocking can be applied on a 4×4 sample grid for luma, the worst-case complexity in terms of number of multiply-accumulate (MAC) operations per sample for luma deblocking has not been changed compared to HEVC. The worst-case is still deblocking of 8×8 samples blocks. The reasons for not increasing the worst-case complexity for luma are twofold. First, linear interpolation is employed in the long-tap deblocking filter design. The long-tap deblocking filter and the associated long-tap spatial activity decision are only applied for large block sizes. Second, when a segment of a block boundary is filtered and at least one side of the boundary has a block side size orthogonal to the boundary equal to 4, both S_P and S_Q are restricted to 1. This limitation only uses the first spatial activity condition in Eq. (16). For chroma, the worst-case deblocking complexity is increased. First, S_P and S_Q are both increased from 1 to 3. Second, an additional long-tap chroma spatial activity decision for each chroma component is needed.

The line buffer requirements for CTU-based deblocking in VVC are the same as in HEVC. This is achieved by limiting the number of lines accessed by the deblocking above the CTU when deblocking a horizontal CTU boundary. The limitation is 4 and 2 lines for luma and chroma, respectively.

In HEVC deblocking, CTU-based processing can be performed independently on 8×8 samples block units [35].

In VVC, the chroma and luma deblocking can be performed independently on block units of 8×8 and 16×16 samples, respectively.

V. Luma Mapping With Chroma Scaling

Unlike ALF and DBF, which have been extensively studied in standards prior to VVC, LMCS is a new coding tool only present in VVC. LMCS consists of a luma mapping (LM) part and a chroma scaling (CS) part. LM remaps the luma code values and CS allows flexible adjustment between luma and chroma signals. LM aims at improving the coding efficiency by reallocating the luma code values of the input video signal within the complete codeword range. For example, in a video signal that conforms to the ITU-R BT.2020-2 [7] or ITU-R BT.2100-2 [8] specification, only luma code values between 64 to 940 are allowed for a 10-bit narrow-range video. Moreover, a video clip may use only a narrow range of luma code values. For example, a 1,000 cd/m^2 high dynamic range (HDR) ITU-R BT.2100 PQ video occupies approximately 75% of the total allowed code values. Such inefficient codeword utilization allows for coding performance improvements via remapping. CS aims at re-balancing the impact of luma remapping on the relative luma/chroma coding bit costs. The flexible adjustment between luma and chroma signal can be achieved by enabling or disabling chroma scaling at the sequence or picture level or further adjusting chroma scaling by applying a chroma scaling offset.

A. Development of LMCS

The design of LMCS (originally referred to as "in-loop reshaper") originated from an earlier proposal of an out-of-loop reshaper. The out-of-loop reshaper applied forward and backward reshaping (or mapping) as out-of-loop pre-processing and post-processing to HEVC Main 10 to improve the subjective coding performance of high dynamic range/wide color gamut (HDR/WCG) PQ videos [50]. The out-of-loop reshaper, in various forms, was adopted to the Exploratory Test Model (ETM) [51] developed during the MPEG exploration phase of HDR video coding.

The first version of the in-loop reshaper was proposed as a response to the call for evidence (CfE) for video compression with capability beyond HEVC [52]–[54], and it was applied only to HDR PQ content. The proposed in-loop reshaper included an up-to-32 pieces linear mapping for the 10-bit luma component and luma-based chroma QP offsets to compensate for the change of the luma signal. A refined in-loop reshaper was proposed in response to the call for proposals (CfP) for video compression beyond HEVC and then included in a JVET Core Experiment (CE) [55]. The subsequent development focused on improvements to support both SDR and HDR (PQ and HLG) content and on implementation simplifications. In the initial design, chroma inverse quantization had to wait for the completion of luma decoding. To alleviate this decoder pipeline dependency issue, an alternate design using chroma residue scaling was proposed [56]. In [57], SDR and HDR were both supported, and the architecture was further simplified by performing intra prediction in the reshaped sample domain and inter prediction in the original sample domain to remove the latency issue in the intra prediction loop. Furthermore, the luma mapping look-up-table was reduced to 16 pieces; the reshaper syntax elements were inserted in the slice header (SH); and chroma scaling was disabled for separate luma/chroma tree to avoid pipeline latency concerns. The resulting in-loop reshaper design was eventually adopted in VVC and renamed as LMCS.

Subsequent contributions further fine-tuned the design. These contributions include: 1. signalling LMCS model parameters in the APS instead of the SH [58] to manage the temporal dependencies; 2. implicitly deriving chroma residue scaling factors from previously reconstructed neighboring luma samples to solve luma-chroma dependent issue [59]; and 3. optionally signalling CS offsets to fine tune the balance between luma and chroma coding performance [60]. The final LMCS design in VVC [62] combines all these contributions.

B. LMCS Coding Tool in VVC

The decoding architecture with LMCS is illustrated in Fig. 11. The upper part of the figure illustrates the CS process, while the lower part of the figure illustrates the LM process. LMCS maps the luma code values of an input video signal from the original (unmapped) sample domain to the mapped sample domain. Thus, the appropriate transformation of the sample values between the two domains may be required. As shown in Fig. 11, the processes in the mapped sample domain (gray-shaded blocks) include inverse quantization (Q^{-1}), inverse transform (T^{-1}), luma intra prediction (Intra Prediction) and summing the luma prediction with the luma residue values (Reconstruction). The processes in the original sample domain include in-loop filters (deblocking, SAO, ALF), inter prediction, chroma intra prediction, summing chroma prediction with the chroma residue values and storage of pictures in a DPB. *Forward luma mapping* which maps the luma code values from the original sample domain to the mapped sample domain, *inverse luma mapping* which maps the luma code values from the mapped sample domain back to the original sample domain, and *Chroma Scaling*, which determines a chroma scaling factor and scales the chroma residue values according to the scaling factor, are the new processes (gray-dotted blocks) introduced by LMCS.

In the decoder, the following steps are performed for LM: a) inverse quantization and inverse transform are applied to the decoded luma transform coefficients to produce the luma residues in the mapped sample domain, Y'_{res}; b) reconstructed luma sample values in the mapped sample domain, Y'_{r}, are obtained by summing Y'_{res} with the corresponding predicted luma values in the mapped sample domain, Y'_{pred} (for intra prediction, Y'_{pred} is directly obtained by performing intra prediction in mapped sample domain, while for inter prediction, the predicted luma values in original sample domain, Y_{pred}, are first obtained by motion compensation using reference pictures from the DPB, and then forward luma mapping is applied to produce the luma values in the mapped sample domain, Y'_{pred}); and c) the reconstructed values, which is the sum of Y'_{pred} and Y'_{res}, is then inverse-mapped and processed

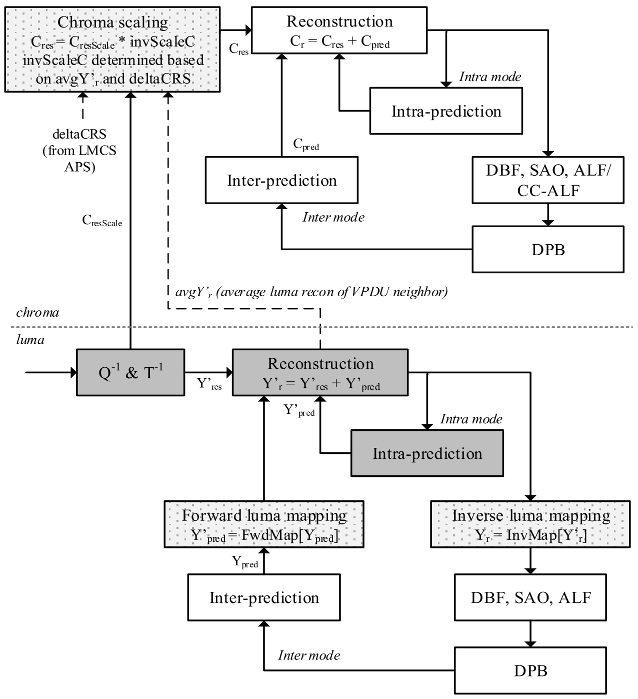


Fig. 11. LMCS decoder diagram.

by other in-loop filters before being stored in the DPB (in the original sample domain).

For CS, the following steps are performed in decoding: a) inverse quantization and inverse transform processes are applied to the decoded chroma transform coefficients to produce chroma residue-scaled values, $C_{res\text{Scale}}$; b) the chroma residue values, C_{res}, are obtained by multiplying $C_{res\text{Scale}}$ by the inverse scaling factor $invScaleC$; c) reconstructed chroma sample values, C_r, are obtained by summing C_{res} with the corresponding predicted chroma value C_{pred}. CS is disabled for chroma blocks with area size less than or equal to 4 samples.

C. LMCS Syntax Design and Model Derivation

The LMCS syntax elements are signalled in the APS with aps_parameter_type set equal to 1 (LMCS_APS). Up to 4 LMCS APSs are supported in a coded video sequence (CVS), but only one LMCS APS may be used for a picture. LMCS data contains two parts: 1) syntax related to a piecewise linear model of up to 16 pieces; 2) a CS offset value *deltaCRS,* if the video signal is not monochrome. LMCS can be controlled at sequence level, picture level or slice level to give the encoder the flexibility to fit the video content and balance the complexity and the performance.

As depicted in Fig. 11, luma mapping uses a forward mapping function, *FwdMap*, and a corresponding inverse mapping function, *InvMap*. These two mapping functions are represented by a piecewise linear model which can be derived from the syntax elements signalled in LMCS APS. The input codeword range of the piecewise linear forward mapping function is uniformly sampled into 16 pieces of same length *OrgCW*. For example, for a 10-bit input video, each of the 16 pieces contains $OrgCW = 64$ input codewords. For each piece of index i, the number of output (mapped) codewords is defined as $binCW[i]$. $binCW[i]$ is determined at the encoding process. The difference between $binCW[i]$ and $OrgCW$ is signalled in LMCS APS. The slopes $scaleY[i]$ and $invScaleY[i]$ of the functions *FwdMap* and *InvMap* are respectively derived as:

$$scaleY[i] = \frac{binCW[i]}{OrgCW} \tag{21}$$

$$invScaleY[i] = \frac{OrgCW}{binCW[i]} \tag{22}$$

The CS applies a forward scaling to the chroma residue with factor $scaleC$ at the encoder and a corresponding inverse scaling with factor $invScaleC$ at the decoder. The value of $invScaleC$ is determined by the number of mapped codewords $binCW[i]$ in the corresponding piece and a chroma scaling offset value, *deltaCRS*, which is signalled in LMCS APS. To reduce the pipeline latency, a single average luma value, $avgY'_r$ is computed for all the coding blocks within the current virtual pipeline data unit (VPDU), which consists of non-overlapping 64 × 64 luma samples. $avgY'_r$ is computed as the average of a fixed number of top and left reconstructed luma samples neighboring the VPDU. The index i of the piece to which $avgY'_r$ belongs is then identified. The value of $invScaleC$ is derived as:

$$invScaleC[i] = \frac{OrgCW}{binCW[i] + deltaCRS} \tag{23}$$

D. LMCS Encoder Parameters Estimation

The VTM software encoder implements two algorithms to estimate LMCS parameters for HDR PQ videos and SDR/HDR HLG videos, respectively. For HDR PQ videos, the algorithm is designed to optimize for weighted PSNR (wPSNR), the luma mapping curve being derived directly based on the weights used in wPSNR [53], [54]. For SDR/HDR HLG, the algorithm is designed to optimize for PSNR. The basic idea of the encoder algorithm for SDR and HDR HLG videos is to assign more luma codewords to spatially smooth areas than non-smooth areas [61], [62]. The mapping curve is estimated only at IRAP pictures and signalled in the APS associated with the IRAP picture. The flexible LMCS design allows encoder to limit or disable LM and/or CS based on the QP values and the picture statistics. To be specific, the VTM encoder allows 3 adaptations: slice adaptation, rate or QP adaptation, and chroma adaptation.

- Slice adaptation: the slice activation of LMCS using the estimated LMCS parameters at IRAP picture can be made according to the following options: 1) for all intra and subsequent inter slices; 2) only for subsequent slices belonging to pictures with TemporalID = 0; 3) only for subsequent inter slices. For example, if the average spatial variance of the picture in the mapped domain exceeds the average spatial variance of the picture in the original (non-mapped) domain by a set of predetermined thresholds, LMCS is either disabled for intra slices, or, alternatively, enabled only for slices belonging to pictures with TemporalID = 0. Otherwise, LMCS is enabled for all slices.

- Rate/QP adaptation: LMCS can be disabled for QP less than or equal to 22 but enabled for higher QPs to preserve fidelity in a picture at higher bitrate.
- Chroma adaptation: CS can be enabled or disabled based on the relative average spatial variance of the luma and chroma components of the picture. If the ratio of the average chroma variance to the average luma variance exceeds a predetermined threshold value, CS is disabled, otherwise CS is enabled.

VI. Experimental Results

This section presents individual experimental results of ALF, CC-ALF, SAO, DBF and LMCS. Cumulative results combining these tools are also provided.

A. Test Conditions

The experiments are conducted using the VTM-9.0 [67] and, for all in-loop filter tools, using the common test conditions (CTC) defined for SDR content [63]. In addition, CC-ALF and LMCS are also tested using the CTC defined for HDR content [64]. CC-ALF results for non-4:2:0 "sensor-generated content" [68] are also reported.

The coding efficiency is measured by the BD-rate variations [65], [66] for each luma and chroma component and for a combined YUV component using a weighted combined per-sequence average PSNR, as suggested in [66]. The weighted PSNR is derived according to Eq. (24), (25) and (26) for 4:2:0, 4:2:2 and 4:4:4 chroma formats, respectively:

$$PSNR\_YUV_{seq} = \frac{6 * PSNR\_Y_{seq} + PSNR\_U_{seq} + PSNR\_V_{seq}}{8} \tag{24}$$

$$PSNR\_YUV_{seq} = \frac{2 * PSNR\_Y_{seq} + PSNR\_U_{seq} + PSNR\_V_{seq}}{4} \tag{25}$$

$$PSNR\_YUV_{seq} = \frac{PSNR\_Y_{seq} + PSNR\_U_{seq} + PSNR\_V_{seq}}{3} \tag{26}$$

The CTC define a set of sequences covering a wide range of resolutions and use cases. They specify the four following configurations:

- All Intra (AI): only intra-prediction is used;
- Random Access (RA): intra pictures are used at certain time intervals, bi-prediction is enabled and inter-prediction can use preceding and future pictures in display order;
- Low Delay B (LDB): only the first picture uses all intra, bi-prediction is enabled and inter-prediction can only use preceding pictures in display order;
- Low Delay P (LDP): same as LDB but only uni-directional inter-prediction is used.

TABLE III
Experimental Results of ALF

Config	Y	U	V	YUV	EncT	DecT
AI	-2.31%	-2.73%	-3.27%	-2.47%	101%	107%
RA	-4.35%	-3.62%	-3.84%	-4.23%	102%	110%
LDB	-4.20%	-6.79%	-5.83%	-4.61%	102%	109%
LDP	-5.96%	-7.51%	-6.69%	-6.18%	104%	109%

B. ALF Performance

The results are summarized in Table III. The anchor is VTM-9.0 with ALF disabled for both luma and chroma and the test is VTM-9.0. ALF achieves 2.47% YUV BD-rate reduction in AI configuration and over 4.2% YUV BD-rate reduction in all the other configurations. When ALF is applied to a picture, the objective quality of this reconstructed picture is improved. If this reconstructed picture is used as a reference frame by another frame, the energy of prediction error is reduced resulting in better coding performance. However, an intra-frame coded picture cannot use filtered sample values as predictors. Therefore, the BD-rate reduction in RA, LDB and LDP configurations is higher than that in AI configuration. EncT and DecT are encoding and decoding time ratios between test and anchor. In VTM-9.0, ALF increases EncT and DecT by around 4% and 10% respectively. This implies that ALF has a bigger impact on DecT than on EncT. When encoding a frame, VTM searches for the best combinations of different coding modes, such as block partitioning, intra modes, inter modes and transforms. Rate and distortion of each combination are calculated by performing prediction, transform, quantization, entropy coding, inverse quantization and inverse transform. On the other hand, in the encoder ALF filtering is only applied once when reconstructed picture is generated. Fast distortion estimation introduced in Section II.F is applied when deriving filter coefficients and deciding on which filter set should be applied to a given CTB.

In addition, improvements brought by the different techniques employed by ALF such as sample difference clipping (Eqs. (4)-(6), luma sub-block filter level adaptation (Section II.B) and CTB level adaptation (Section II.C) are analyzed. Results obtained for RA configuration by disabling each of these techniques separately are summarized in Table IV. In each row of Table IV, the anchor is VTM-9.0 with the technique indicated in the first column disabled and the test is VTM-9.0. For each technique, EncT and DecT increase by less than 3% with over 0.6% YUV BD-rate reduction.

In VTM-9.0, although ALF encoder is designed to improve objective quality, subjective quality improvement can also be observed. In example in Fig. 12, ALF reduces both ringing artifacts (marked with the black circles) and blocking artifacts (marked with the white circles).

C. CC-ALF Performance

In this experiment, CC-ALF is disabled in the anchor and enabled in the test. For all QP values, the VTM is configured to consider only Lagrangian cost minimization to derive CC-ALF parameters (CCALFQpTh parameter set to 100). For CC-ALF,

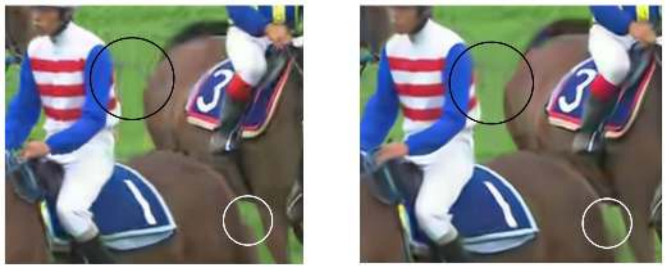


Fig. 12. RaceHorses, LDB, QP 37: (left) ALF off, (right) ALF on.

TABLE IV
EXPERIMENTAL RESULTS OF ALF SUB-TOOLS

	Y	U	V	YUV	EncT	DecT
Clipping	-0.71%	-0.94%	-1.18%	-0.79%	102%	101%
Sub-block adaptation	-1.34%	-0.02%	0.01%	-1.05%	102%	102%
CTB adaptation	-0.45%	-1.03%	-1.32%	-0.62%	101%	102%

the traditional BD-rate measurement tends to show a loss in luma but a gain in chroma, since the additional bits spent by CC-ALF are used to improve the chroma quality while the luma quality remains nearly the same. The test results for CC-ALF are shown in Table V and contain three parts. The first part shows the results for the case where the encoder targets PSNR gains and the content is in YCbCr color space with ITU-R BT.1886 Opto-Electrical Transfer Function (OETF) [69]. In the second part, the encoder again targets PSNR gains, while the content is in RGB color space with ITU-R BT.1886 OETF. The third part is related to PQ 4:2:0 content and shows results where the encoder targets a weighted PSNR metric. For PQ 4:2:0, the input $PSNR\_Y_{seq}$, $PSNR\_U_{seq}$ and $PSNR\_V_{seq}$ to Eq. (24), correspond to the per-sequence wPSNRY, wPSNRU and wPSNRV [64] output by VTM. From the first part, the combined YUV BD-rate gain for CC-ALF is between 1.6% to 6.4% in every encoder configuration, for both SDR and HLG content, amongst all chroma formats. For the content using RGB color space, the CC-ALF gain is between 0.7% to 4.3%. Since CC-ALF operates on chroma channels, the results in Table V demonstrate higher gains for 4:4:4 and 4:2:2 content. This is because the ratio of chroma samples to luma samples is larger compared to that in 4:2:0 content. Also, worth noting is the fact that although the chroma location types of HLG and PQ content is different, the coding efficiency gains for the two are similar, since CC-ALF does not impose any symmetry restrictions on its filter coefficients. The increased decoding time ratio for CC-ALF is reflective of the fact that its implementation in VTM does not include any Single Instruction, Multiple Data (SIMD) optimizations.

D. SAO Performance

Table VI reports the BD-rate improvement of enabling SAO in VTM-9.0 with/without ALF and CC-ALF, for SDR content. The anchor in each case corresponds to SAO disabled. When ALF and CC-ALF are enabled, the average YUV BD-rate

TABLE V
EXPERIMENTAL RESULTS OF CC-ALF

Part 1: Encoder optimizing PSNR

Source	Y	U	V	YUV	EncT	DecT
All Intra (AI)						
SDR 4:2:0	0.18%	-9.88%	-8.30%	-1.67%	100%	103%
HLG 4:2:0	0.15%	-15.25%	-15.14%	-1.96%	100%	104%
SDR 4:2:2	0.09%	-5.99%	-10.32%	-3.00%	100%	105%
SDR 4:4:4	0.16%	-2.09%	-8.06%	-2.74%	100%	109%
Random Access (RA)						
SDR 4:2:0	0.16%	-13.09%	-12.33%	-2.69%	100%	101%
HLG 4:2:0	0.15%	-20.04%	-20.15%	-2.70%	101%	102%
SDR 4:2:2	0.17%	-12.91%	-15.36%	-4.84%	100%	104%
SDR 4:4:4	0.17%	-4.62%	-13.13%	-4.59%	101%	106%
Low Delay B (LDB)						
SDR 4:2:0	0.17%	-16.64%	-11.39%	-2.56%	101%	102%
HLG 4:2:0	0.14%	-27.64%	-16.66%	-2.81%	99%	101%
SDR 4:2:2	-0.02%	-16.71%	-18.95%	-6.24%	100%	103%
SDR 4:4:4	-0.28%	-6.10%	-16.65%	-6.30%	100%	107%
Low Delay P (LDP)						
SDR 4:2:0	0.17%	-16.86%	-11.67%	-2.57%	100%	101%
HLG 4:2:0	0.18%	-28.36%	-16.99%	-2.83%	98%	100%
SDR 4:2:2	-0.02%	-17.16%	-19.43%	-6.37%	100%	104%
SDR 4:4:4	-0.18%	-6.26%	-17.13%	-6.41%	100%	107%

Part 2: Encoder optimizing PSNR – SDR RGB 4:4:4

Cfg.	G	B	R	GBR	EncT	DecT
AI	0.07%	-0.39%	-1.92%	-0.72%	100%	107%
RA	-0.22%	-2.08%	-4.65%	-2.36%	100%	109%
LDB	-1.74%	-3.18%	-6.51%	-3.77%	100%	110%
LDP	-2.10%	-3.76%	-7.31%	-4.35%	99%	111%

Part 3: Encoder optimizing wPSNR[64] (PQ YCbCr 4:2:0)

Cfg.	Y	U	V	YUV	EncT	DecT
AI	0.23%	-12.64%	-25.88%	-1.99%	100%	104%
RA	0.60%	-13.97%	-30.38%	-2.43%	100%	103%
LDB	1.24%	-21.73%	-32.57%	-2.07%	96%	99%
LDP	1.38%	-21.03%	-32.00%	-1.83%	96%	99%

gain of enabling SAO in four test conditions is 0.2%. When ALF and CC-ALF are disabled, SAO not only achieves 1.3% YUV BD-rate gains in average but also provides subjective benefits, as shown in Fig. 13. The BD-rate improvement of enabling SAO is partially overlapped with ALF. However, considering that SAO is well deployed in HEVC and the required computation complexity is very low, SAO is kept in VVC to allow using SAO to further improve both objective and subjective quality, especially when ALF is disabled.

E. DBF Performance

Table VII shows the objective BD-rate gain by enabling the DBF for various CTC configurations. For AI configuration the DBF increases the bitrate by 0.6%. For other configurations, the DBF decreases the bitrate between 1% and 1.6%. The long-tap deblocking filter, which is the main difference between the HEVC and the VVC deblocking filter, provides similar bitrate at the same objective quality as can be seen in Table VIII. It can be observed that the objective performance from enabling the DBF does not provide the same amount of

TABLE VI
EXPERIMENTAL RESULTS OF SAO

SDR Config		Y	U	V	YUV
Enable ALF & CC-ALF	AI	-0.01%	-0.14%	-0.20%	-0.04%
	RA	-0.09%	-0.23%	-0.34%	-0.13%
	LDB	-0.12%	-0.43%	-1.19%	-0.24%
	LDP	-0.12%	-0.70%	-1.35%	-0.28%
Disable ALF & CC-ALF	AI	-0.33%	-0.33%	-0.67%	-0.36%
	RA	-0.80%	-0.77%	-0.82%	-0.80%
	LDB	-0.92%	-1.84%	-3.39%	-1.22%
	LDP	-2.48%	-2.55%	-4.14%	-2.64%

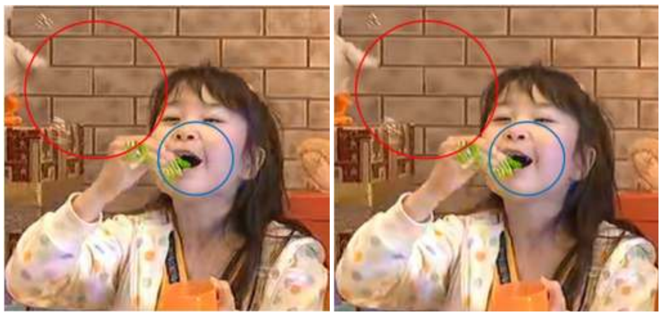


Fig. 13. Screenshot illustrating visual quality improvement from SAO for PartyScene @ 1.13Mbps with VTM-9.0 and ALF disabled in LDP case: (left) SAO off, (right) SAO on.

TABLE VII
EXPERIMENTAL RESULTS OF DBF

Config	Y	U	V	YUV
AI	0.60%	-0.30%	-0.71%	0.36%
RA	-1.04%	-1.10%	-1.29%	-1.07%
LDB	-1.07%	-1.90%	-2.23%	-1.27%
LDP	-1.58%	-2.13%	-2.56%	-1.73%

improvement for VVC as for HEVC [35]. The main reason is that VVC includes ALF, and in VTM ALF is optimized to produce a picture as close to the source picture as possible. ALF takes part of the objective gain from the DBF, because in VTM, the DBF is optimized mainly to increase subjective quality, not to reduce the MSE between the source picture and the reconstructed picture. The tuning of the t_C and β parameters can also be used to achieve better objective and/or subjective performance. To reduce the amount of smoothing of a color component, the value of t_C and β can be reduced for that component. To increase the amount of smoothing, the value of t_C and β can be increased. Fig. 14 shows the visual quality improvement using the DBF for AI, RA and LDB cases, respectively. It can be observed that the VVC DBF can effectively attenuate the blocky artifacts, as highlighted in red and blue circles.

F. LMCS Performance

Table IX reports the BD-rate improvement of enabling LMCS for SDR content in AI, RA, LDB and LDP cases. The anchor in each case corresponds to LMCS disabled. The interaction of LM and CS components can be appreciated

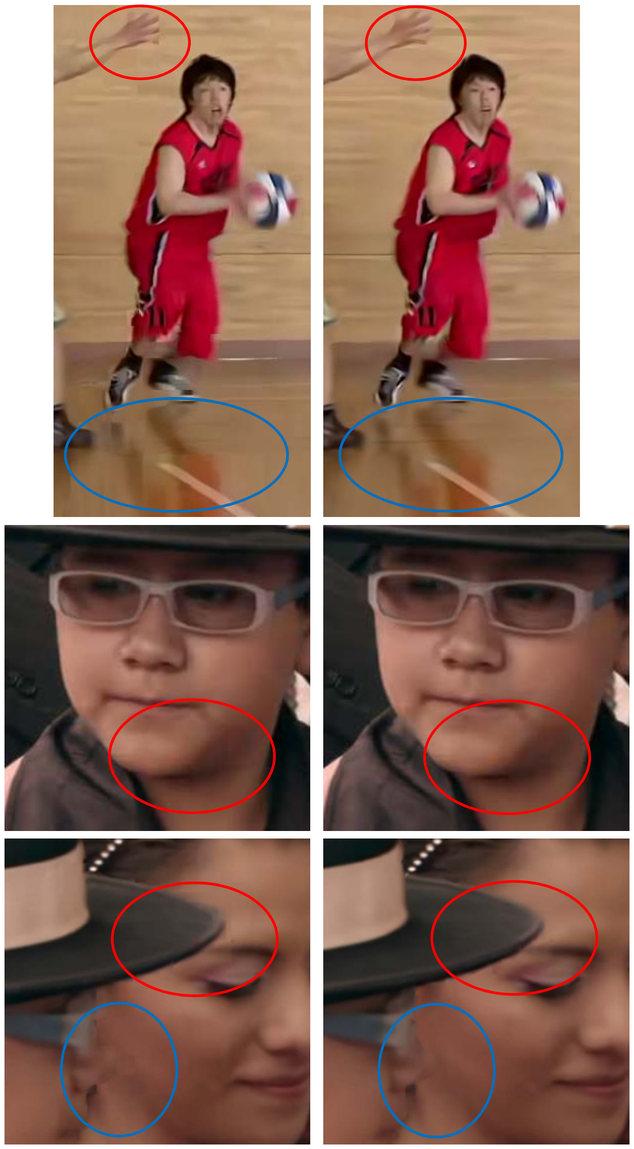


Fig. 14. (left) DBF off, (right) DBF on. Top: Basketball Drive, LDB, QP 37; Middle: Tango, AI, QP 37; Bottom: Tango, RA, QP 37.

TABLE VIII
EXPERIMENTAL RESULTS OF LONG-TAP DBF SUB-TOOL

Config	Y	U	V	YUV
AI	-0.02%	-0.23%	-0.22%	-0.07%
RA	0.01%	-0.43%	-0.38%	-0.09%
LDB	0.06%	-0.50%	-0.85%	-0.08%
LDP	0.04%	-0.72%	-1.21%	-0.15%

by comparing the data shown in Table X to the data shown in Table IX (SDR RA). When CS is completely disabled, LM results in a chroma coding performance loss, as indicated by results of Table X (LM only, CS disabled). When CS is enabled and scaling factors are derived implicitly, chroma performance loss is reduced, as indicated by results of Table X (LMCS with *deltaCRS* $= 0$). When CS is enabled and chroma scaling factors are explicitly signalled in the LMCS APS, LMCS results in both luma and chroma coding performance gain, as indicated by the results of Table IX (SDR RA).

TABLE IX
EXPERIMENTAL RESULTS OF LMCS: SDR 4:2:0

Config	Y	U	V	YUV	EncT	DecT
AI	-0.94%	-0.29%	-0.60%	-0.86%	101%	102%
RA	-1.34%	-1.02%	-0.88%	-1.28%	105%	102%
LDB	-0.90%	0.33%	0.21%	-0.68%	105%	101%
LDP	-0.97%	0.06%	0.12%	-0.77%	105%	101%

TABLE X
EXPERIMENTAL RESULTS OF LMCS SUB-TOOLS: SDR RA

	LM only, CS disabled			LMCS with deltaCRS=0		
	Y	U	V	Y	U	V
Overall	-1.68%	4.36%	4.09%	-1.52%	1.50%	1.35%

TABLE XI
EXPERIMENTAL RESULTS OF LMCS: HDR-HLG RA

	Y	U	V	YUV
HDR HLG	-0.97%	-0.59%	-0.64%	-0.90%

TABLE XII
EXPERIMENTAL RESULTS OF LMCS: HDR-PQ RA

DE100	PSNR-L100	wPSNRY	wPSNRU	wPSNRV	wYUV
-1.05%	-1.35%	-1.05%	-0.39%	-0.19%	-1.02%

TABLE XIII
EXPERIMENTAL RESULTS OF OVERALL IN-LOOP FILTER: SDR 4:2:0

Config	Y	U	V	YUV
AI	-4.17%	-14.10%	-14.78%	-6.35%
RA	-9.54%	-18.72%	-18.43%	-11.64%
LDB	-8.67%	-22.96%	-19.90%	-11.27%
LDP	-13.60%	-25.56%	-22.70%	-15.68%

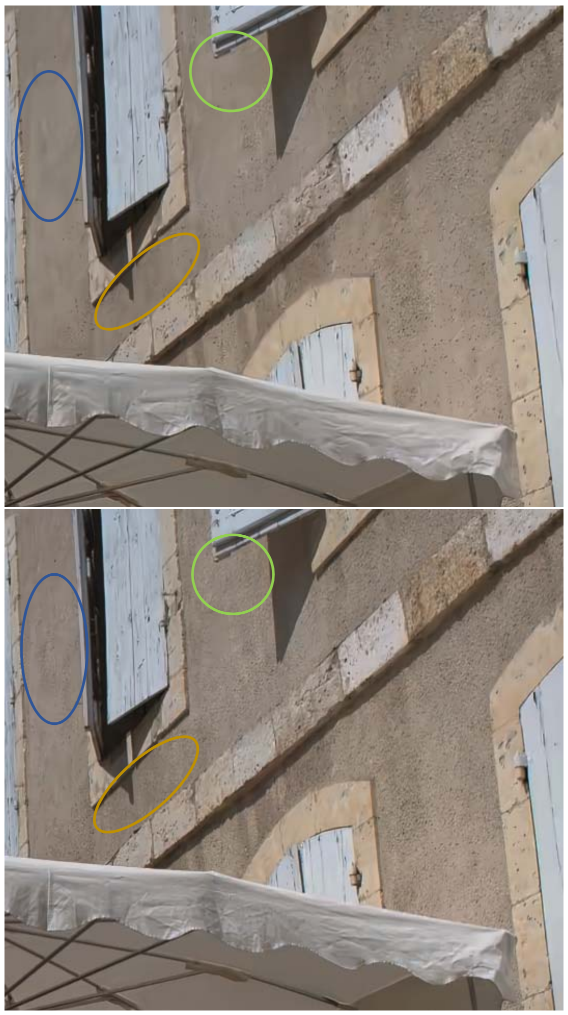


Fig. 15. Screenshot illustrating visual quality improvement from LMCS for Market @ 1Mbps with VTM-9.0: (top) LMCS off, (bottom) LMCS on.

The performance of LMCS for HDR sequences are shown in Table XI for HDR HLG sequences and Table XII for HDR PQ sequences in RA configuration. For HDR HLG sequences, LMCS is evaluated with the same objective metrics as those for SDR. For HDR PQ sequences, the objective metrics for evaluation are DE100, PSNRL100, wPSNRY, wPSNRU and wPSNRV [64]. The anchor is LMCS disabled and luma-dependent quantization enabled.

To show subjective improvement, LMCS is configured to use an adaptive curve. The design is based on [53], where a parametric model estimates the subjective importance of each piece of the piecewise linear model and pre-defines codeword allocation for luma intervals based on piece importance. Fig. 15 shows a snapshot of a PQ sequence (Market). With LMCS on, the wall texture is better preserved under same bitrate (1Mbps).

G. In-Loop Filter Overall Performance

Table XIII reports the overall performance of in-loop filters in VVC for SDR 4:2:0 content. The reference is VTM-9.0 with LMCS, ALF, CC-ALF, SAO, and DBF off. The test is VTM-9.0 with all in-loop filters activated. The set of in-loop filters brings a substantial gain for all encoder configurations and for all three color components and for the PSNR-YUV case. Higher gains are observed in inter configurations, with 11.64% PSNR-YUV gains in RA configuration, and highest gains obtained in LDP (15.68% in PSNR-YUV), which tends to show that the VVC in-loop filters can very efficiently compensate for the less efficient temporal prediction.

VII. CONCLUSION

VVC defines five different in-loop filters that bring significant compression efficiency improvements, both in terms of objective and subjective quality. Three tools are mostly designed for reducing coding artifacts: DBF, SAO and ALF. SAO is identical to its HEVC design while DBF is an enhanced version of the HEVC DBF, with the usage of long-tap filters and a more flexible deblocking control including a luma level-dependent control. ALF is a new tool bringing substantial coding gains for both luma and chroma components. The ALF gain comes from the precise block classification of ALF allowing a fine tuning of the ALF offsets per block class. The fourth tool, CC-ALF, takes advantage of the correlation between luma and chroma for improving the signal fidelity and providing objective gains for chroma components. The fifth tool, LMCS is in a different category: It improves the coding efficiency by adaptively exploiting the video signal range. In RA configuration using the JVET CTC, the set of in-loop filtering tools brings an average PSNR-Y, U and V cumulative BD-rate gain of 9.54%, 18.72%, 18.43% for SDR content. When considering PSNR-YUV, the average BD-rate gain is 11.64%. Gains of higher range are observed for the LDP inter configuration with 15.68% PSNR-YUV average BD-rate gain.

The design of the in-loop filter tools has been made with particular care on their complexity and implementation for software and hardware platforms. Parallelization support has also been carefully considered, which is of high interest for new video applications using very high picture resolutions such as 4K and 8K and beyond, virtual reality with 360° video, and cloud gaming applications.

References

[1] *Versatile Video Coding*, Standard Recommendation ITU-T H.266, Aug. 2020.

[2] *Advanced Video Coding*, Standard Recommendation ITU-T H.264, May 2019.

[3] *High Efficiency Video Coding*, Standard Recommendation ITU-T H.265, Nov. 2019.

[4] C.-Y. Tsai *et al.*, "Adaptive loop filtering for video coding," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 934–945, Dec. 2013.

[5] Y.-W. Chen *et al.*, *Description of SDR, HDR and 360° Video Coding Technology Proposal by Qualcomm and Technicolor—Low and High Complexity Versions*, document JVET-J0021, 10th JVET Meeting, San Diego, CA, USA, Apr. 2018.

[6] 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.

[7] *Parameter Values for Ultra-High Definition Television Systems for Production and International Programme Exchange*, Standard Recommendation ITU-R BT.2020-2, 2015.

[8] *Image Parameter Values for High Dynamic Range Television for Use in Production and International Programme Exchange*, Standard Recommendation ITU-R BT.2100-2, 2018.

[9] M. Karczewicz, N. Shlyakhov, N. Hu, V. Seregin, and W.-J. Chien, *CE2.4.1.4: Reduced filter shape size for ALF*, document JVET-K0371, 11th JVET Meeting, Ljubljana, Slovenia, Jul. 2018.

[10] Y.-C. Su, C.-Y. Chen, Y.-W. Huang, and S.-M. Lei, *CE2-Related: Reduction of Bits for ALF Coefficient Fractional Part*, document JVET-L0083, 12th JVET Meeting, Macao, China, Oct. 2018.

[11] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Comput. Vis.*, Jan. 1998, pp. 839–846.

[12] J. Taquet, P. Onno, C. Gisquet, and G. Laroche, *CE5: Results of Tests CE5-3.1, CE5-3.2, CE5-3.3 and CE5-3.4 on Non-Linear Adaptive Loop Filter*, document JVET-N0242, 18th JVET Meeting, Geneva, Switzerland, Apr. 2019.

[13] M. Karczewicz, L. Zhang, W.-J. Chien, and X. Li, *EE2.5: Improvements on Adaptive Loop Filter*, document JVET-C0038, 3rd JVET Meeting, Geneva, Switzerland, Jun. 2016.

[14] S.-C. Lim, J. Kang, H. Lee, J. Lee, and H. Y. Kim, *CE2: Subsampled Laplacian Calculation (Test 6.1, 6.2, 6.3, and 6.4)*, document JVET-L0147, 12th JVET Meeting, Macao, China, Oct. 2018.

[15] J. Taquet, P. Onno, C. Giquet, and G. Laroche, *CE5-4: Alternative Luma Filter Sets and Alternative Chroma Filters for ALF*, document JVET-O0090, 15th JVET Meeting, Gothenburg, Sweden, Jul. 2019.

[16] N. Hu, V. Seregin, H. E. Egilmez, and M. Karczewicz, *CE5: Coding Tree Block Based Adaptive Loop Filter (CE5-4)*, document JVET-N0415, 14th JVET Meeting, Geneva, Switzerland, Mar. 2019.

[17] A. M. Kotra, S. Esenlik, B. Wang, H. Gao, and J. Chen, *Non-CE: Loop Filter Line Buffer Reduction*, document JVET-M0301, 13th JVET Meeting, Marrakech, Morocco, Jan. 2019.

[18] K. Andersson, J. Ström, Z. Zhang, and J. Enhorn, *Fix for ALF Virtual Boundary Processing*, document JVET-Q0150, 17th JVET Meeting, Brussels, Belgium, Jan. 2020.

[19] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[20] K. Misra, F. Bossen, and A. Segall, *Cross-Component Adaptive Loop Filter for Chroma*, document JVET-O0636, 15th JVET Meeting, Gothenburg, Sweden, Jul. 2019.

[21] K. Misra, F. Bossen, and A. Segall, "On cross component adaptive loop filter for video compression," in *Proc. Picture Coding Symp. (PCS)*, Ningbo, China, Nov. 2019, pp. 1–5, doi: 10.1109/PCS48520.2019.8954547.

[22] J. Dong, Y. He, and Y. Ye, *Chroma Enhancement for ILR Picture*, document JCTVC-L0059, 12th JCT-VC Meeting, Geneva, Switzerland, Jan. 2013.

[23] X. Li, J. Chen, W. Pu, and M. Karczewicz, *Non-SCE4: Simplification of Chroma Enhancement for Inter Layer Reference Picture Generation*, document JVET-M0253, 13th JCTVC Meeting, Incheon, KR, Apr. 2013.

[24] J. Li and C. S. Lim, *AHG16/Non-CE5: Cross Component ALF Simplification*, document JVET-P0173, 16th JVET Meeting, Geneva, Switzerland, Oct. 2019.

[25] J. Taquet, P. Onno, C. Gisque and, and G. Laroche, *Non-CE5: CC-ALF Filtering Simplification*, document JVET-P0330, 16th JVET Meeting, Geneva, Switzerland, Oct. 2019.

[26] K. Misra, F. Bossen, and A. Segall, *CE5-Related: Reducing Multiplier Count in CC-ALF*, document JVET-P0468, 16th JVET Meeting, Geneva, Switzerland, Oct. 2019.

[27] N. Hu, J. Dong, V. Seregin, and M. Karczewicz, *CE5-Related: Multiplication Removal for Cross Component Adaptive Loop Filter*, document JVET-P0557, 16th JVET Meeting, Geneva, Switzerland, Oct. 2019.

[28] Z. Zhang, J. Ström, and K. Andersson, *CE5-Related: On the CC-ALF Filtering Process*, document JVET-Q0165, 17th JVET Meeting, Brussels, Belgium, Jan. 2020.

[29] O. Chubach, C.-Y. Chen, C.-Y. Lai, T.-D. Chuang, Y.-W. Huang, and S.-M. Lei, *CE5-Related: On CC-ALF Modifications Related to Coefficients and Signaling*, document JVET-Q0190, 17th JVET Meeting, Brussels, Belgium, Jan. 2020.

[30] K. Misra *et al.*, *CC-ALF: Integrated Text for the Cross Component Adaptive Loop Filter*, document JVET-Q0795, 17th JVET Meeting, Brussels, Belgium, Jan. 2020.

[31] N. Hu, V. Seregin, and M. Karczewicz, *AHG16: Line Buffer Problem of CC-ALF for 4:2:2 and 4:4:4 Sequences*, document JVET-R0233, 18th JVET Meeting, by Teleconference, Apr. 2020.

[32] X. W. Meng, X. Zheng, S. S. Wang, and S. W. Ma, *AHG 10: One-Pass CCALF*, document JVET-R0225, 18th JVET Meeting, by Teleconference, Apr. 2020.

[33] W.-S. Kim *et al.*, "Cross-component prediction in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 6, pp. 1699–1708, Jun. 2020.

[34] K. Zhang, J. Chen, L. Zhang, X. Li, and M. Karczewicz, "Enhanced cross-component linear model for chroma intra-prediction in video coding," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3983–3997, Aug. 2018.

[35] A. Norkin *et al.*, "HEVC deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1746–1754, Dec. 2012.

[36] A. Ichigaya, S. Iwamura, and S. Nemoto, *CE11: Luma-Adaptive Deblocking Filter (CE11.2.2)*, document JVET-L0414, 12th JVET Meeting, Macao, China, Oct. 2018.

[37] K. Andersson, Z. Zhang, and R. Sjöberg, *CE11: Deblocking of Sub-Block Boundaries for Luma (CE11.3.2)*, document JVET-L0074, 12th JVET Meeting, Macao, China, Oct. 2018.

[38] K. Andersson *et al.*, *CE11-2.1: Deblocking for 4xN, Nx4 and 8xN and Nx8 Block Boundaries not Aligned With 8x8 Grids*, document JVET-N0098, 14th JVET Meeting, Geneva, Switzerland, Mar. 2019.

[39] K. Misra, A. Segall, M. Ikeda, T. Suzuki, K. Andersson, and J. Enhorn, *Non-CE11: On ISP Transform Boundary Deblocking*, document JVET-N0473, 14th JVET Meeting, Geneva, Switzerland, Mar. 2019.

[40] M. Ikeda *et al.*, *CE11.1.6, CE11.1.7 and CE11.1.8: Joint Proposals for Long Deblocking From Sony, Qualcomm, Sharp, Ericsson*, document JVET-M0471, 13th JVET Meeting, Marrakech, Morocco, Jan. 2019.

[41] K. Andersson and J. Enhorn, *CE1: CE1-1.1 to CE1-1.3: Fixes for Long Luma Deblocking Filter Decision*, document JVET-Q0054, 17th JVET Meeting, Brussels, Belgium, Jan. 2020.

[42] A. M. Kotra, S. Esenlik, B. Wang, H. Gao, and E. Alshina, *CE5-1.3: Unified Design for Longer Tap Deblocking Line Buffer Reduction*, document JVET-P0081, 16th JVET Meeting, Geneva, Switzerland, Oct. 2019.

[43] K. Andersson *et al.*, *CE5-2.1 and CE5-2.2: Deblocking on 4x4 Sample Grids*, document JVET-O0060, 15th JVET Meeting, Gothenburg, Sweden, Jul. 2019.

[44] A. Norkin, *CE11-Related: On Deblocking Tc Table*, document JVET-L0410, 12th JVET Meeting, Macao, China, Oct. 2018.

[45] K. Andersson and J. Enhorn, *Non-CE5: Deblocking tC Table Defined for 10-bit Video*, document JVET-O0159, 15th JVET Meeting, Gothenburg, Sweden, Jul. 2019.

[46] S.-T. Hsiang and S. Lei, *CE7.3.2: Extension of Quantization Parameter Value Range*, document JVET-K0251, 11th JVET Meeting, Ljubljana, Slovenia, Jul. 2018.

[47] K. Misra, P. Cowan, A. Segall, Y. Morigami, M. Ikeda, and T. Suzuki, *Non-CE5: On Chroma Line Selection for Gradient Computation in Deblocking*, document JVET-O0637, 15th JVET Meeting, Gothenburg, Sweden, Jul. 2019.

[48] J. Xu, Y.-K. Wang, L. Zhang, W. Zhu, K. Zhang, and H. Liu, *AHG9: On Deblocking Control Parameters*, document JVET-Q0121, 17th JVET Meeting, Brussels, Belgium, Jan. 2020.

[49] V. Baroncini, A. Norkin, and A. M. Kotra, *Subjective Assessment of CE11 Proposals*, document JVET-L0611, 12th JVET Meeting, Macao, China, Oct. 2018.

[50] D. Baylon *et al.*, *Response to Call for Evidence for HDR and WCG Video Coding: Arris, Dolby and InterDigital*, document m36264, 112th MPEG Meeting, Warsaw, Poland, Jul. 2015.

[51] K. Minoo, T. Lu, P. Yin, L. Kerofsky, D. Rusanovskyy, and E. Francois, *Description of the Exploratory Test Model (ETM) for HDR/WCG Extension of HEVC*, document JCTVC-W0092, 10th JCT-VC Meeting, San Diego, CA, USA, Apr. 2016.

[52] X. Xiu *et al.*, *Description of SDR, HDR and 360° Video Coding Technology Proposal by InterDigital Communications and Dolby Laboratories*, document JVET-J0015, 10th JVET Meeting, San Diego, CA, USA, Apr. 2018.

[53] X. Xiu *et al.*, "A unified video codec for SDR, HDR, and 360° video applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1296–1310, May 2020.

[54] E. Francois, C. A. Segall, A. M. Tourapis, P. Yin, and D. Rusanovskyy, "High dynamic range video coding technology in responses to the joint call for proposals on video compression with capability beyond HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1253–1266, May 2020.

[55] T. Lu, F. Pu, P. Yin, W. Husak, S. McCarthy, and T. Chen, *CE12: HDR In-loop Reshaping (CE12-5, pp. ,–12, 12-7 and 12-8)*, document JVET-K0308, 11th JVET Meeting, Ljubljana, Slovenia, Jul. 2018.

[56] T. Lu, F. Pu, P. Yin, W. Husak, S. McCarthy, and T. Chen, *CE12-2: HDR In-Loop Reshaping*, document JVET-L0245, 12th JVET Meeting, Macao, China, Oct. 2018.

[57] T. Lu, F. Pu, P. Yin, W. Husak, S. McCarthy, and T. Chen, *CE12: Mapping Functions (Test CE12-1 and CE12-2)*, document JVET-M0427, 13th JVET Meeting, Marrakech, Morocco, Jan. 2019.

[58] W. Wan *et al.*, *AHG17: Design for Signalling Reshaper Model*, document JVET-N0805, 14th JVET Meeting, Geneva, Switzerland, Mar. 2019.

[59] J. Chen *et al.*, *Non-CE2: Unification of Chroma Residual Scaling Design*, document JVET-O1109, 15th JVET Meeting, Gothenburg, Sweden, Jul. 2020.

[60] E. François, F. Galpin, K. Naser, and P. D. Lagrange, *AHG7/AHG15: Signalling of Corrective Values for Chroma Residual Scaling*, document JVET-P0371, 16th JVET Meeting, Geneva, Switzerland, Oct. 2019.

[61] J. Chen, Y. Ye, and S. Kim, *Algorithm Description for Versatile Video Coding and Test Model 7 (VTM 7)*, document JVET-R2002, 18th JVET Meeting, Online Meeting, Apr. 2020.

[62] T. Lu *et al.*, "Luma mapping with chroma scaling in versatile video coding," in *Proc. IEEE Data Compress. Conf. (DCC)*, Snowbird, UT, USA, Mar. 2020, pp. 193–202.

[63] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, *JVET Common Test Conditions and Software Reference Configurations for SDR Video*, document JVET-N1010, 14th JVET Meeting, Geneva, Switzerland, Mar. 2019.

[64] A. Segall, E. François, W. Husak, S. Iwamura, and D. Rusanovskyy, *JVET Common Test Conditions and Evaluation Procedures for HDR/WCG Video*, document JVET-P2011, 16th JVET Meeting, Geneva, Switzerland, Oct. 2019.

[65] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD Curves*, document VCEG-M33, 13th VCEG Meeting, Texas, USA, Mar. 2001.

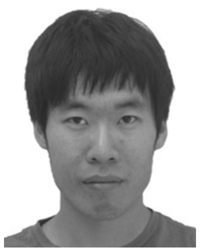
[66] *Working Practices Using Objective Metrics for Evaluation of Video Coding Efficiency Experiments*, Standard ITU-T Technical paper HSTP-VID-WPOM, Jul. 2020. [Online]. Available: https://www.itu.int/pub/T-TUT-ASC-2020-HSTP1

[67] *VTM-9.0*. Accessed: Jun. 2020. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tags/VTM-9.0

[68] Y.-H. Chao, Y.-C. Sun, J. Xu, and X. Xu, *JVET Common Test Conditions and Software Reference Configurations for Non-4:2:0 Colour Formats*, document JVET-R2013, 18th JVET Meeting, Online Meeting, Apr. 2020.

[69] *Reference Electro-Optical Transfer Function for Flat Panel Displays Used in HDTV Studio Production*, Standard Recommendation ITU-R BT,1886, Mar. 2011.

Marta Karczewicz received the M.Sc. and Ph.D. degrees from the Tampere University of Technology, Finland, in 1994 and 1997, respectively. From 1996 to 2006, she was with Nokia. In 2006, she joined Qualcomm, where she is currently a Vice President with the Multimedia Research and Development Group. Since 1998, she has been active participant in H.264/AVC, HEVC, and VVC video codecs development within MPEG and ITU-T standardization forums. While in Qualcomm, she has also contributed to projects related to hardware video encoder implementation and universal bandwidth compression.



Nan Hu joined Qualcomm Inc., in 2016. He is currently a Staff Engineer with Qualcomm, San Diego, CA, USA. His research interest includes image and video coding. He is an active contributor to VVC standardization.

Jonathan Taquet received the M.S. degree in computer science from the University of La Rochelle, La Rochelle, France, in 2007, and the Ph.D. degree in signal processing and telecommunications from the University of Rennes 1, Rennes, France, in 2011. He was with the Canon Research Centre France at the time this non-linear ALF extension was proposed for VVC standardization. His research interests include signal processing, analysis and synthesis, multimedia compression and communication, and machine learning.

Ching-Yeh Chen received the B.S. degree in electrical engineering and the Ph.D. degree in electronics engineering from National Taiwan University (NTU), Taipei, Taiwan, in 2002 and 2006, respectively. He is currently a Senior Department Manager with the Multimedia Technology Development Division, MediaTek Inc. He started attending international video coding standard meetings held by MPEG/VCEG in 2010. He is also an active contributor. His current research interests include image and video coding, image and video processing, and related hardware architectures.

Kiran Misra received the B.E. degree in electronics engineering from Mumbai University, India, in 1998, and the M.S. and Ph.D. degrees in electrical and computer engineering from Michigan State University (MSU), East Lansing, MI, USA, in 2002 and 2010, respectively. In 2010, he joined Sharp Laboratories of America Inc., as a Post-Doctoral Researcher, where he is currently a Principal Researcher with the Video Coding Group. He has actively contributed to video coding standardization within MPEG/VCEG. He was also the Vice-Chair of the Ad-hoc Group on Accessibility during ATSC-3.0 standardization. His research interests include video coding and delivery.

Kenneth Andersson received the M.Sc. degree in computer science and engineering from Luleå University in 1995 and the Ph.D. degree from Linköping University in 2003. In 1994, he started to work at Ericsson, where he is currently a Senior Specialist. Since 2005, he has been active in video coding standardization in ITU-T and ISO/IEC for development of High-Efficiency Video Coding (HEVC) and Versatile Video Coding (VVC). His main research interests include image and video signal processing, and video coding.

Peng Yin received the B.E. degree in electrical engineering from the University of Science and Technology of China in 1996 and the Ph.D. degree in electrical engineering from Princeton University in 2002. She worked with Thomson Inc./Technicolor from 2002 to 2010. She is currently working as the Director of Imaging Applied Research, Dolby Laboratories, Inc. She is actively involved in MPEG/VCEG video coding related standardizations. Her research interests include image/video processing and compression. She received the IEEE Circuits and Systems Society Best Paper Award for her article in the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY in 2003.

Taoran Lu received the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2010. She is currently a Senior Staff Engineer with the Advanced Technology Group, Dolby Laboratories, Inc., Sunnyvale, CA, USA. Her research interests include image and video compression and transmission, next-generation video coding, image and video analysis, and computer vision.

Edouard François received the Engineering degree from the ENST de Bretagne, France, and the Ph.D. degree in computer science from University de Rennes 1, France. He was with Canon Research, France, from 2011 to 2013. He has been with InterDigital since 2019. He had been working with Technicolor (formerly Thomson) for around 20 years. He is currently a Principal Scientist with InterDigital and leading a research team focused on video compression. He has been participating to several standardization groups, including MPEG and VCEG. His main research interests include video motion analysis and processing, video coding, and High Dynamic Range video.

Jie Chen received the B.S. and M.S. degrees in information and communication engineering from Zhejiang University, Hangzhou, China, in 2009 and 2012, respectively. He is currently with XG Labs, Damo Academy, Alibaba Group, Beijing, China. Before joining Alibaba in 2019, he was with the Beijing Samsung Telecom Research and Development Center, Samsung Electronics, from 2012 to 2018. He has been actively involved in the development of various video coding standards, including VVC standard, AVS3 standard, and AVS2 standard. His research interests include image and video coding and processing.