Subsampled Sum-Modified-Laplacian for Adaptive Loop Filter in Versatile Video Coding

SUNG-CHANG LIM\textsuperscript{1,2}, MYUNGJUN KIM\textsuperscript{1}, JUNGWON KANG\textsuperscript{2}, (Member, IEEE), AND YUNG-LYUL LEE\textsuperscript{1}, (Senior Member, IEEE)

\textsuperscript{1}Department of Computer Engineering, Sejong University, Seoul 05006, South Korea
\textsuperscript{2}Communication and Media Research Laboratory, Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, South Korea

Corresponding author: Yung-Lyul Lee (yllee@sejong.ac.kr)

This work was supported by the Institute for Information and Communications Technology Planning and Evaluation (IITP) funded by the Korea Government (MSIT) (Development and Standardization of 6DoF Volumetric Video Coding Technology) under Grant 2019-0-00012.

ABSTRACT This paper proposes a Subsampled Sum-Modified-Laplacian (SSML) operator for the block classification of the Adaptive Loop Filter (ALF) in Versatile Video Coding (VVC). The VVC Test Model (VTM)-2.0 includes Geometry transformation-based ALF (GALF) with $4 \times 4$ block classification, a single $7 \times 7$ Luma diamond-shaped filter, and spatial adaptation at the Coding Tree Block (CTB) level to improve the coding efficiency of VVC. However, in the $4 \times 4$ block classification, 1-D Modified-Laplacian (ML) values for various directions are calculated at all sample positions within an $8 \times 8$ window, and these are summed to derive the gradients for the corresponding directions. For a CTB where the Wiener filter is applied, every $4 \times 4$ Luma block within the CTB must calculate Sum-Modified-Laplacian (SML) operations, which results in increased computational complexity of the decoder. Therefore, four different subsampling patterns based on $4 \times 4$ block classification for the SSML operator that calculate the 1-D ML values at subsampled sample positions within the $8 \times 8$ window are proposed as attempts to reduce the computational complexity of the block classification process in the ALF of VTM-2.0. As all of the proposed subsampling patterns reduce the number of sample positions needed to calculate the 1-D ML operations by half, the experimental results demonstrate that the proposed method achieves total decoding time reductions in the range of 2% to 3%, while the coding gain of the ALF is maintained.

INDEX TERMS Adaptive loop filter (ALF), block classification, in-loop filter, versatile video coding (VVC), video coding.

I. INTRODUCTION

Video coding is one of the most important key technologies for video applications, such as those used in mobile devices, personal computers, Ultra High Definition (UHD) televisions, video conferencing, video streaming, remote screen sharing, and cloud gaming. The state-of-the-art video coding standard is Versatile Video Coding (VVC) [1], which was developed through the collaborative efforts of the Joint Video Experts Team (JVT) of the ITU-T WP3/16 Video Coding Experts Group (VCEG) and the ISO/IEC JTC 1/SC 29/WG 11 Moving Picture Experts Group (MPEG) and finalized in July 2020. VVC is the successor to block-based hybrid video codecs such as the Advanced Video Coding (AVC) standard [2], [3] and the High Efficiency Video Coding (HEVC) standard [4], [5]. The main target application areas for VVC include UHD video, High Dynamic Range (HDR) and Wide Color Gamut (WCG) video, screen content video, and 360° omni-directional video, as well as conventional Standard Definition (SD) and High Definition (HD) videos. VVC provides coding efficiency improvements of about 40% [6] over HEVC for UHD test sequences in terms of the objective quality measure of BD (Bjøntegaard Delta)-rate [7], [8]. However, VVC requires about two times higher complexity than HEVC in terms of the decoding runtime due to the advance in its coding tools [6]. Compared to its predecessors, VVC has many dominant coding tools which provide coding efficiency improvements of more than 1%, such
as Cross-Component Linear Model (CCLM), affine motion model, Adaptive Motion Vector Resolution (AMVR), Multiple Transform Selection (MTS), Dependent Quantization (DQ), Luma Mapping with Chroma Scaling (LMCS), and Adaptive Loop Filter (ALF) [9], [10]. The ALF in particular is one of the most significant coding tools in VVC since it achieves BD-rate saving of more than 4% in “Random Access Main 10” (RA) configuration [10].

Including the ALF, three kinds of in-loop filters, which improve both the objective and subjective aspects of visual quality for reconstructed pictures, are supported in VVC. The first in-loop filter is the Deblocking Filter (DF), which smooths block boundaries to suppress block artifacts. The second one is the Sample Adaptive Offset (SAO), which adaptively compensates for the sample distortion by adding an offset. The last one is the ALF, which minimizes the distortion between the original samples and the reconstructed samples by using a Wiener filter. Here, the Wiener filter has been used as an optimal linear filter to improve the degraded picture. The ALF is typically composed of a classification process and a filtering process. In the classification process, each sample or block is classified into a class according to a certain classification rule. In addition, one set of Wiener filter coefficients is optimized to the corresponding one or more classes in the encoder to minimize the distortion between the original samples and the reconstructed samples. Further, the optimized Wiener filter coefficients are encoded into the bitstream as side information so that the decoder can use the same filter coefficients in the decoding process. At both the encoder and decoder, the optimized Wiener filters are applied to the reconstructed signal in the filtering process to improve both the objective and subjective visual qualities. As a result, there exists a rate-distortion tradeoff of the ALF between the rate of side information and the improved visual quality.

In VVC, an ALF method based on Geometry transformation-based ALF (GALF) [11]–[13] is included as a third in-loop filter. In addition, the VVC Test Model (VTM) 2.0 reference software [14] includes the GALF with 4 × 4 block classification, a single 7 × 7 Luma diamond-shaped filter, and spatial adaptation at the Coding Tree Block (CTB) level [15]. Although the ALF in VTM-2.0 provides a significant coding gain, it also involves high computational complexity in the block classification process. In the 4 × 4 block classification of the ALF in VTM-2.0, 1-D (1-Dimensional) Modified-Laplacian (ML) values for each of the horizontal, vertical, main diagonal, and anti-diagonal directions are calculated at all sample positions within an 8 × 8 window, and these are summed to derive the gradients for the corresponding four directions. For a CTB where the Wiener filter is applied, the sums of the 1-D ML values, called Sum-Modified-Laplacian (SML), should be calculated in every 4 × 4 Luma block within the CTB, and those processes require many operations in the block classification process.

Therefore, in this paper, a Subsampled Sum-Modified-Laplacian (SSML) operator for the 4 × 4 block classification is proposed to reduce the computational complexity associated with measuring the local characteristics, i.e., the directionality and the activity, while supporting four kinds of gradient directions in the block classification process of the ALF in VVC. To derive the directionality and the activity of the 4 × 4 block, each of the gradients of the horizontal, vertical, main diagonal, and anti-diagonal directions are calculated using the proposed SSML operator. This paper also investigates the impacts of four different subsampling patterns for the proposed SSML operator on the coding efficiency and the computational complexity. The 4 × 4 block classification based on the SSML operator presented in this paper was proposed in the JVET standardization [16]–[18] to provide the reduction of the decoding complexity for VTM-2.0. It should be noted that the second subsampling pattern of the proposed SSML operator in this paper was adopted into the VVC Working Draft (WD) 3 [19] and VTM-3.0 [20], [21] in the 12th JVET meeting. Finally, it is included in the Final Draft International Standard (FDIS) of VVC.

The remainder of this paper is structured as follows. In Section II, a summary of the ALF proposals and a description of the GALF, which is the base method of the proposed method, are described. Next, a description of the 4 × 4 block classification based on the proposed SSML operator and an analysis of the computational complexity between VTM-2.0 and the proposed method are presented in Section III. Then, the experimental results of the proposed method are provided in Section IV. Finally, Section V concludes the study.

II. PREVIOUS WORKS

A. SUMMARY OF ALF PROPOSALS FOR VIDEO CODING STANDARDS

For many years, the ALF has attracted interest as a promising research topic to improve the coding efficiency of video codecs. When VCEG investigated coding tools that could potentially provide improved performances beyond AVC during the Key Technology Areas (KTA) activity, two different kinds of ALF method using the Wiener filter in the decoding loop [22], [23] were first proposed at the same VCEG meeting. In [22], reconstructed samples were classified into two groups according to their local characteristics in a slice, and one of the two different filters was then applied to the corresponding group. However, since this method classified the characteristics of all samples in the slice into only two classes, the improvement in the coding efficiency was limited. Another drawback of these methods [22], [23] is that the filter was optimized in each slice without considering the spatial adaptation of filter usage. The spatial adaptation was later considered in Block-based ALF (BALF) [24] and Quadtree-based ALF (QALF) [25]. In the BALF, a slice was divided into blocks of the same size, and the block size information was encoded into the bitstream for each slice. Additional information was also encoded to indicate whether or not each block was filtered. The QALF introduced a quadtree-based block partitioning method to support the adaptivity of the filter usage for each block. Accordingly, two flags were encoded to indicate the quad-tree partitioning information.
and the block-level filter usage. To improve the coding efficiency of QALF by increasing the number of classes, another ALF method [26] that classified the reconstructed samples into many classes according to its own characteristic of texture and edge information was proposed. In [26], among up to 16 filter sets, one filter set was selected based on a measure of the local texture activity of the reconstructed samples, called the value of the SML operator. In [27], the SML operator was first introduced to provide a local measure of the quality of image focus that was obtained by calculating the sum of the absolute second derivative values in the horizontal and vertical directions within a window. Further, the 1-D ML operator was defined as the absolute value of the 1-D Laplacian operator to detect the directional edge. Due to the effectiveness of the SML operator, sample-level classification based on the SML operator has been extended to block-level classification through many proposals for video coding standards.

Compared to sample-level classification, 4 × 4 block-level classification with SML-based activity metrics was proposed in [28] in an attempt to improve the coding efficiency and reduce the computational complexity during the development of HEVC. According to [28], each 4 × 4 block was classified into one of 16 classes depending on the SML-based activity. For one or more classes, the Wiener filter was optimized to reduce the distortion of the classified blocks. However, the main drawback of the method was that the 1-D ML operators had to be calculated at all sample positions in the 4 × 4 block. In [29], [30], to further reduce the computational complexity of the 4 × 4 block classification, the authors suggested removing the summation of 1-D ML values within a 3 × 3 window and 1-D ML operations at every second sample position in both the horizontal and vertical directions in the 4 × 4 block for the ALF design in HEVC. However, since the 1-D ML operators were calculated at the sample positions within a block instead of a window, which is an overlapping area consisting of the current block and its neighboring samples, texture and edge information on the neighboring samples could not be employed. It also led to coding loss due to the small number of samples used to calculate the 1-D ML operators. Further, since the 1-D ML values of other directions such as the diagonal directions were not considered, this method was limited in terms of the edge and texture information it could capture. Various other methods related to the ALF were proposed to improve both the coding efficiency and the computational complexity during the HEVC standardization, and these were adopted into the HEVC Test Model (HM); however, the ALF was not included in the HEVC profiles due to its relatively high computational complexity compared to other coding tools [31].

After the HEVC standardization was finalized, the Joint Exploration Test Model (JEM) was developed by VCEG and MPEG to study the potential need for standardization of the next generation video coding standard with improved compression capability over HEVC [32], [33]. The ALF was again included in JEM to evaluate its potential coding efficiency and the possibility of its future adoption in the next generation video coding standard. The ALF method in JEM-7.0 [34], so-called Geometry transformation-based ALF (GALF), performed a 2 × 2 block classification to classify each 2 × 2 block into one of 25 classes based on the directionality and the activity of the 2 × 2 block. Further, 1-D ML operators for two diagonal directions, such as main diagonal and anti-diagonal directions, were added to those for the horizontal and vertical directions to calculate the directionality of the 2 × 2 block. Before each 2 × 2 block was filtered using the filters optimized according to the corresponding classes, geometric transformations such as diagonal flipping, vertical flipping, and 270° rotation were applied to the filter coefficients based on the various gradients calculated from the sum of the 1-D ML values. For the filtering of each 2 × 2 block, one of the three diamond-shaped filters with sizes of 9 × 9, 7 × 7, and 5 × 5 was selected to filter the Luma component, while a 5 × 5 diamond-shaped filter was used for the Chroma component [11]–[13].

During the development of VVC, the GALF was first included in Benchmark Set (BMS)-1.0 [35], which was built on top of VTM and included many promising video coding tools. Later, 4 × 4 block classification, a single 7 × 7 Luma diamond-shaped filter, and spatial adaptation at the Coding Tree Block (CTB) level [15] on top of the GALF were adopted into VTM-2.0. Although the ALF in VTM-2.0 provided a significant coding gain, it also had high computational complexity in the block classification process. Specifically, in the 4 × 4 block classification, the 1-D ML operators still needed to be calculated at all sample positions within an 8 × 8 window, including the 4 × 4 block and its surrounding two lines of top, bottom, left, and right, for four directions: the horizontal, vertical, main diagonal, and anti-diagonal directions.

B. DESCRIPTION OF GALF

In this section, the design of GALF related to the proposed method is described. The GALF has major features, including 2 × 2 block classification with two additional diagonal gradients, geometry transformations of the filter coefficients according to the calculated gradients, and predictive coding of the filter coefficients from fixed filters to improve coding efficiency compared to JEM-1.0 [36]. Fig. 1 shows an example block diagram of the VTM decoder, where the GALF is placed at the end of the in-loop filtering process. As shown in the figure, the block classification process and the filtering process in the GALF are performed on the reconstructed samples processed by the DF (Deblocking Filter) and the SAO (Sample Adaptive Offset). In addition, the output signal of the GALF is stored in the Decoded Picture Buffer (DPB) for display purposes or to be referenced by succeeding pictures.

The 2 × 2 block classification is performed based on the 1-D ML values that are the results of the 1-D ML operations to efficiently capture the local texture characteristics of each 2 × 2 block; i.e., the 2 × 2 block classification depends on the directionality based on each sum of 1-D ML values.
Assuming the horizontal gradient derivation process is conducted to derive the directionality:

(2-Dimensional) ML values corresponding to the sum of 1-D horizontal ML values and 1-D vertical ML values.

First, in the block classification process, the following derivation process is conducted to derive the directionality: Assuming the horizontal gradient \(G_H\), the vertical gradient \(G_V\), the main diagonal (135°) gradient \(G_{D0}\), and the anti-diagonal (45°) gradient \(G_{D1}\), four gradient values are obtained by summing the 1-D ML values, where the 1-D ML values, \(H_{k,l}, V_{k,l}, D0_{k,l},\) and \(D1_{k,l}\) are calculated at all sample positions within a 6 x 6 window for the corresponding horizontal, vertical, main diagonal, and anti-diagonal directions, respectively, as in Eqs. (1)-(4).

\[
G_H = \sum_{k=-2}^{i+3} \sum_{l=-2}^{i+3} H_{k,l},
\]

\[
H_{k,l} = [2 \times R(k, l) - R(k - 1, l - 1) - R(k - 1, l + 1) - R(k, l + 1)]
\]

\[
G_V = \sum_{k=-2}^{i+3} \sum_{l=-2}^{i+3} V_{k,l},
\]

\[
V_{k,l} = [2 \times R(k, l) - R(k - 1, l - 1) - R(k, l + 1) - R(k + 1, l - 1)]
\]

\[
G_{D0} = \sum_{k=-2}^{i+3} \sum_{l=-2}^{i+3} D0_{k,l},
\]

\[
D0_{k,l} = [2 \times R(k, l) - R(k - 1, l - 1) - R(k - 1, l + 1) - R(k, l + 1) - R(k + 1, l - 1)]
\]

\[
G_{D1} = \sum_{k=-2}^{i+3} \sum_{l=-2}^{i+3} D1_{k,l},
\]

\[
D1_{k,l} = [2 \times R(k, l) - R(k - 1, l + 1) - R(k + 1, l - 1) - R(k, l - 1) - R(k - 1, l + 1)]
\]

where indices \(i, j\) indicate the coordinates of the top-left sample position in the 2 x 2 shaded block and \(R(i,j)\) indicates the reconstructed sample at coordinate \((i, j)\).

Fig. 2 shows the sample positions, where the 1-D ML operators are calculated within the 6 x 6 window. In the figure, the shaded regions indicate the current 2 x 2 block, and \(H, V, D0,\) and \(D1\) mean the sample positions where the 1-D ML operators are calculated for the horizontal, vertical, main diagonal, and anti-diagonal directions, respectively. In Fig. 2, the 1-D ML operators are calculated at all sample positions within the 6 x 6 window for the four directions. Then, by using the obtained four gradient values, the ratio of the maximum and minimum of the horizontal and vertical gradients, denoted by \(R_{H,V}\) in Eq. (5), as well as the ratio of the maximum and minimum of the main diagonal and anti-diagonal gradients, denoted by \(R_{D0,D1}\) in Eq. (6), are derived.

\[
R_{H,V} = \frac{G_{H,V}^{\text{MAX}}}{G_{H,V}^{\text{MIN}}},
\]

where \(G_{H,V}^{\text{MAX}} = \max(G_H, G_V)\), \(G_{H,V}^{\text{MIN}} = \min(G_H, G_V)\) (5)

\[
R_{D0,D1} = \frac{G_{D0,D1}^{\text{MAX}}}{G_{D0,D1}^{\text{MIN}}},
\]

where \(G_{D0,D1}^{\text{MAX}} = \max(G_{D0}, G_{D1})\), \(G_{D0,D1}^{\text{MIN}} = \min(G_{D0}, G_{D1})\) (6)

Moreover, by comparing \(R_{H,V}\) and \(R_{D0,D1}\) with two thresholds \(T_1\) and \(T_2\), the directionality \(D\) represented by five
representative direction values within the range of 0 to 4 are derived using Eq. (7).

\[
D = \begin{cases} 
0 & \text{if } R_{H,V} \leq T_1 \&\& R_{D0,D1} \leq T_1 \\
1 & \text{if } R_{H,V} > T_1 \&\& R_{H,V} > R_{D0,D1} \&\& R_{H,V} \leq T_2 \\
2 & \text{if } R_{H,V} > T_1 \&\& R_{H,V} > R_{D0,D1} \&\& R_{H,V} > T_2 \\
3 & \text{if } R_{D0,D1} > T_1 \&\& R_{H,V} \leq R_{D0,D1} \&\& R_{D0,D1} \leq T_2 \\
4 & \text{if } R_{D0,D1} > T_1 \&\& R_{H,V} \leq R_{D0,D1} \&\& R_{D0,D1} > T_2 
\end{cases}
\] (7)

where the derived directionality for each \(2 \times 2\) block has the following meaning: \(D = 0\) for texture block, \(D = 1\) for strong horizontal or vertical edge block, \(D = 2\) for weak horizontal or vertical edge block, \(D = 3\) for strong main diagonal or anti-diagonal edge block, and \(D = 4\) for weak main diagonal or anti-diagonal edge block.

Secondly, the activity \(A\) of the \(2 \times 2\) block based on the sum of the 2-D ML values is derived using Eq. (8), then quantized to the range of 0 to 4, inclusively.

\[
A = \sum_{k=-2}^{3} \sum_{l=-2}^{3} (H_{i+k,j+l} + V_{i+k,j+l})
\] (8)

Finally, depending on \(D\) and the quantized activity \(A_Q\), the classification index \(C\) for the \(2 \times 2\) block is derived as in Eq. (9).

\[
C = 5 \times D + A_Q
\] (9)

It should be noted that, in the \(2 \times 2\) block classification, all samples in the \(2 \times 2\) block share the same \(C\). In contrast to the previous work mentioned in Section II, \(A\), in the GALF, each \(2 \times 2\) block is classified into one of 25 classes. Further, a Wiener filter could be optimized to one or more classes in the encoder, and up to 25 sets of optimized filter coefficients may be encoded into the bitstream. For example, if a diamond-shaped filter with a \(7 \times 7\) size is used, 13 filter coefficients per set are encoded. For the Chroma component, the block classification is not applied, so only one filter is used for each component ofCb or Cr.

In the previous ALF designs, after each block was classified in the block classification process, the filter coefficients corresponding to the class were used to filter the reconstructed sample in the filtering process. Meanwhile, in the GALF, one more stage, which performs a geometry transformation of the filter coefficients, is added between the block classification process and the filtering process. To more accurately distinguish the directionality in the case that more than one \(2 \times 2\) blocks are classified into the same \(C\) in Eq. (9), four different kinds of geometry transformation, including no transformation, diagonal flip, vertical flip, and \(270^\circ\) rotation, are supported in the GALF. The geometry transformations are applied to the filter coefficients depending on the results of the comparison of \(G_V\) and \(G_H\) and the comparison of \(G_{D0}\) and \(G_{D1}\) for each \(2 \times 2\) block. Specifically, if \(G_H\) is larger than \(G_V\) and \(G_{D0}\) is larger than \(G_{D1}\), then diagonal flipping is applied to the filter coefficients; if \(G_V\) is larger than \(G_H\) and \(G_{D1}\) is larger than \(G_{D0}\), then vertical flipping is applied to the filter coefficients; and if \(G_H\) is larger than \(G_V\) and \(G_{D1}\) is larger than \(G_{D0}\), \(270^\circ\) rotation is applied to the filter coefficients. Fig. 3 shows a \(7 \times 7\) diamond-shaped filter and its three geometric transformed versions of diagonal flip, vertical flip, and \(270^\circ\) rotation. In the figure, the shaded squares illustrate the circular symmetric diamond-shaped filter and \(F_i\) represents the \(i\)-th filter coefficient. Therefore, different filtering results can be obtained by using geometry transformations according to the local gradient values without the overhead associated with encoding additional filter coefficients.

In the filtering process, each \(R(i,j)\) is filtered by the filter coefficients that were selected according to \(C\) and geometry-transformed using the gradients. The filtered reconstructed sample \(R_F(i,j)\) is calculated as shown in Eq. (10), where \(K\) denotes the filter length and \(F(k,l)\) represents the filter coefficient.

\[
R_F(i,j) = \sum_{k=-K/2}^{K/2} \sum_{l=-K/2}^{K/2} F(k,l) \times R(i+k,j+l)
\] (10)

III. PROPOSED METHOD

To reduce the computational complexity while maintaining the coding efficiency compared to VTM-2.0, the following observation is considered in the proposed method. Taking into account the results of the previous works described in Section II, it can be assumed that the more gradient directions and classes are supported in the block classification, the better...
the local characteristics obtained in terms of edge and texture information. As a result, the coding efficiency of the ALF can be improved. Further, the samples neighboring the current block within the window improve the robustness for the local characteristic measure, because more local samples are involved in the block classification. On the other hand, since adjacent samples tend to be highly correlated in a picture [37], it might not be necessary to measure the local characteristics using all samples in a local region consisting of highly correlated samples. It could be more efficient to instead represent the local characteristics with a subset of samples. In addition, ensuring that the positions for the subset of samples in the block are fixed for every block could be more beneficial for both software and hardware implementations. Based on the above observations, four different subsampling patterns based on $4 \times 4$ block classification for the SSML operator that calculate the 1-D ML values on subsampled sample positions within an $8 \times 8$ window while supporting four kinds of gradient directions are proposed in this section. It should be noted that among the several processes of the Galf, including the block classification process, geometry transformation process of filter coefficients, and filtering process, the proposed method focuses solely on the block classification process.

### A. Subsampled Sum-Modified-Laplacean (SSML) Operator

In $4 \times 4$ block classification, the filter coefficients are determined according to the directionality and the activity to sort out the local characteristics of each $4 \times 4$ Luma block in the reconstructed picture. In other words, the classification index corresponding to the filter index for each $4 \times 4$ Luma block is derived based on its directionality and quantized activity, and all samples in the $4 \times 4$ Luma block share the same classification index. To calculate the directionality and the quantized activity for each $4 \times 4$ Luma block, each gradient of the horizontal, vertical, main diagonal, and anti-diagonal directions is calculated by summing the 1-D ML values on all sample positions within the $8 \times 8$ window, in contrast to the $6 \times 6$ window considered in the case of $2 \times 2$ block classification.

Four different subsampling patterns based on $4 \times 4$ block classification for the proposed SSML operator are described in this section. Although using a large subsampling factor can further reduce the computational complexity, it may be accompanied by high coding loss. Considering this, the proposed method keeps the subsampling factors for both the horizontal and vertical directions as small as possible to maintain the coding gain of the ALF.

Fig. 4 shows the subsampled sample positions for the first subsampling pattern of the proposed method for calculating 1-D ML operators, including $H$, $V$, $D_{0}$, and $D_{1}$, within the $8 \times 8$ window used to obtain the gradients, including $G_{H}$, $G_{V}$, $G_{D_{0}}$, and $G_{D_{1}}$, respectively. In the figure, the shaded regions indicate the current $4 \times 4$ block. Further, the sample positions for calculating 1-D ML operators are subsampled along the opposite gradient direction to preserve the edge information to allow for the corresponding gradient direction to be calculated. In other words, to employ consecutive samples according to the same gradient direction to the extent possible, every second line along the opposite gradient direction is excluded from the calculation of 1-D ML operators. This means that for the horizontal gradient calculation, vertical subsampling is used as shown in Fig. 4 (a). For the vertical gradient calculation, horizontal subsampling is used as shown in Fig. 4 (b). In addition, subsampling to the anti-diagonal direction is performed for the main diagonal gradient calculation as shown in Fig. 4 (c). Finally, subsampling along the main diagonal direction is performed for the anti-diagonal gradient calculation as shown in Fig. 4 (d). Therefore, the number of samples used to calculate the 1-D ML operators is reduced by half. The gradients of the proposed first subsampling pattern are calculated as in Eqs. (11)-(14), where the $1$-D ML values for the sample positions that are not included in the subsampled sample positions are set equal to 0.

$$
G_{H} = \sum_{k=-2}^{5} \sum_{l=-1}^{2} H_{l+k,j+2l},
$$

$$
H_{p,q} = |2 \times R(p, q) - R(p - 1, q) - R(p + 1, q)|
$$

$$
G_{V} = \sum_{k=-1}^{5} \sum_{l=-2}^{5} V_{i+2k,j+l},
$$

$$
V_{p,q} = |2 \times R(p, q) - R(p, q - 1) - R(p, q + 1)|
$$

$$
G_{D_{0}} = \sum_{k=-1}^{2} \sum_{l=-1}^{2} \left( D_{0_{l+2k,j+2l}} + D_{0_{l+2k+1,j+2l+1}} \right),
$$

$$
D_{0_{p,q}} = |2 \times R(p, q) - R(p - 1, q - 1) - R(p + 1, q + 1) |
$$
Eqs. (15)-(18).

by summing the 1-D ML values of the sample positions

events of the proposed second subsampling pattern are obtained

gradient directions in a unified way. Accordingly, the gradi-

operator calculation.

are used to subsample the sample positions for the 1-D ML

and third subsampling patterns, different quincunx patterns

are clearly be seen that the subsampled sample positions shown

sampling pattern. Fig. 6 shows the unified subsampled sample

positions of the proposed third subsampling pattern, where

FIGURE 5. Sample positions of the proposed second subsampling pattern.

In this pattern, if both the horizontal and vertical indices of

the samples within the window are equal to even numbers or

odd numbers, then the 1-D ML operations are calculated on

those sample positions.

To investigate the impact on the coding efficiency of a
different quincunx pattern from the proposed second sub-
sampling pattern, the sample positions are subsampled along
the main diagonal direction in the proposed third subsam-
pling pattern. Fig. 6 shows the unified subsampled sample
positions for the proposed third subsampling pattern, where
the subsampled sample positions are shifted by one sample
compared to the proposed second subsampling pattern. It can
clearly be seen that the subsampled sample positions shown
in Fig. 4 (d) are used to subsample the sample positions for
this pattern. Further, the gradients for the horizontal, vertical,
main diagonal, and anti-diagonal directions of the proposed
third subsampling pattern are calculated as in Eqs. (19)-(22),
respectively.

\[
G_{D1} = \sum_{k=-1}^{2} \sum_{l=-1}^{2} (D1_{i+2k+1,j+2l} + D1_{i+2k,j+2l+1}),
\]

\[
D1_{p,q} = |2 \times R(p, q) - R(p - 1, q + 1) - R(p + 1, q - 1)|
\]

(14)

In the second, third, and fourth subsampling patterns of the
proposed method, the 1-D ML operators are calculated on the
fixed subsampled sample positions regardless of the gradient
direction to further simplify the block classification process
as shown in Figs. 5-7. Therefore, the numbers of samples
needed to calculate the 1-D ML operators for the second,
third, and fourth subsampling patterns are also reduced by
half, as shown in Figs. 5-7, respectively. For the second
and third subsampling patterns, different quincunx patterns
are used to subsample the sample positions for the 1-D ML
operator calculation.

Fig. 5 shows the subsampled sample positions for the second
subsampling pattern of the proposed method. This pattern
employs the identical pattern used in Fig. 4 (c) for all
gradient directions in a unified way. Accordingly, the gradi-

events of the proposed second subsampling pattern are obtained

by summing the 1-D ML values of the sample positions
subsampled along the anti-diagonal direction as expressed in
Eqs. (15)-(18).

\[
G_{H} = \sum_{k=-1}^{2} \sum_{l=-1}^{2} (H_{i+2k,j+2l} + H_{i+2k+1,j+2l+1})
\]

(15)

\[
G_{V} = \sum_{k=-1}^{2} \sum_{l=-1}^{2} (V_{i+2k,j+2l} + V_{i+2k+1,j+2l+1})
\]

(16)

\[
G_{D0} = \sum_{k=-1}^{2} \sum_{l=-1}^{2} (D0_{i+2k,j+2l} + D0_{i+2k+1,j+2l+1})
\]

(17)

\[
G_{D1} = \sum_{k=-1}^{2} \sum_{l=-1}^{2} (D1_{i+2k,j+2l} + D1_{i+2k+1,j+2l+1})
\]

(18)
VOLUME 8, 2020

To take advantage of vectorized implementation such as Single Instruction Multiple Data (SIMD), the fourth subsampling pattern is proposed to effectively fit horizontally-aligned samples into the SIMD register. This pattern employs the subsampling method equivalent to Fig. 4 (a), where horizontally consecutive samples are involved in the calculation of 1-D ML operations. Fig. 7 shows the subsampled sample positions used to calculate the 1-D ML operators, and it can be seen that every second row within the 8×8 window is excluded from the calculation of the 1-D ML operators for all gradient directions. Therefore, for the proposed fourth subsampling pattern, the gradients for the horizontal, vertical, main diagonal, and anti-diagonal directions are obtained by summing the 1-D ML values of the sample positions subsampled along the vertical direction, as in Eqs. (23)-(26), respectively.

\[
G_H = \sum_{k=-2}^{2} \sum_{l=-2}^{2} H_{i+k,j+2l}
\]

\[
G_V = \sum_{k=-2}^{2} \sum_{l=-1}^{2} V_{i+k,j+2l}
\]

\[
G_{D0} = \sum_{k=-2}^{2} \sum_{l=-1}^{2} D_{0_{i+k,j+2l}}
\]

\[
G_{D1} = \sum_{k=-2}^{2} \sum_{l=-1}^{2} D_{1_{i+k,j+2l}}
\]

The rest of the derivation process for \( D \) and \( A_Q \) with the calculated gradients is the same as the GALF, because the proposed method only modifies the gradient calculations.

**B. ANALYSIS OF COMPUTATIONAL COMPLEXITY**

This section analyzes and compares the computational complexity of the VTM-2.0 anchor, which includes the GALF with 4×4 block classification, and the 4×4 block classification based on the proposed SSML operator. Table 1 presents a comparison of the computational complexity of the decoder between VTM-2.0 and the proposed method with respect to the number of operations in the ALF processes, including the block classification process and the filtering process, for a given 8×8 block.

For the 4×4 block classification in VTM-2.0, the 1-D ML values, \( H, V, D_0, \) and \( D_1 \), that are calculated at all sample positions within the 8×8 window, are summed to derive the gradients, \( G_H, G_V, G_{D0}, \) and \( G_{D1} \), respectively, for each 4×4 Luma block. As listed in Table 1, for a given 8×8 block, calculating the 1-D ML operators for all four gradient directions requires 1584 additions (36 pieces of 2×2 block × 11 additions per 2×2 block × 4 directions), 576 comparisons (36 pieces of 2×2 block × 4 comparisons per 2×2 block × 4 directions), and 144 shifts (36 pieces of 2×2 block × 4 shifts per 2×2 block). Here, 12×12 window of a given 8×8 block is divided into 36 pieces of 2×2 block for 2×2 block-based operation calculation. Further, 240 additions (4 units of 8×8 window × 15 additions per 8×8 window × 4 directions) are required to sum the calculated 1-D ML values for each 8×8 window of four 4×4 block.

To determine the directionality and the quantized activity with the calculated gradients, 8 additions, 28 comparisons, 8 multiplications, and 20 shifts are needed. Therefore, 1832 additions, 604 comparisons, 8 multiplications, and 164 shifts are required in total for the 4×4 block classification in VTM-2.0 for a given 8×8 block. In addition, the numbers of operations per sample are 28.625 additions, 9.4375 comparisons, 0.125 multiplications, and 2.5625 shifts.

By contrast, in the proposed method, 720 additions (36 pieces of 2×2 block × 5 additions per 2×2 block × 4 directions), 288 comparisons (36 pieces of 2×2 block × 2 comparisons per 2×2 block × 4 directions), and 72 shifts (36 pieces of 2×2 block × 2 shifts per 2×2 block) are required to calculate the 1-D ML operators for all four gradient directions, as the number of sample positions are reduced by half. The operations in the remaining block classification process and filtering for both the Luma and Chroma components are the same as those for VTM-2.0. Therefore, the proposed method requires a total of 968 additions, 316 comparisons, 8 multiplications, and 92 shifts for a given 8×8 block, as shown in Table 1. The numbers of operations per sample are calculated as 15.125 additions, 4.9375 comparisons, 0.125 multiplications, and 1.4375 shifts.

As the number of sample positions is the same for all of the proposed subsampling patterns, the number of operations is also the same for those patterns. Since the number of multiplications is the same for both VTM-2.0 and the proposed method, and is also relatively small compared to those of other operations, only additions, comparisons, shifts can be considered in the comparison of computational complexity.
Therefore, excluding the multiplication count, the proposed method reduces 46.25% of the worst-case operation counts compared to VTM-2.0 for the total operation counts in the block classification. Further, regarding the total operation counts of the ALF, 22.11% of the worst-case operation counts is reduced by using the proposed method.

IV. EXPERIMENTAL RESULTS

To capture the runtime reduction and the BD-rate impact of the proposed method, four tests corresponding to the proposed four different subsampling patterns were conducted. To demonstrate the experimental results of the proposed method more precisely without any interference between the proposed method and other coding tools in the latest VTM, VTM-2.0 was chosen over the latest VTM as the anchor. Therefore, the proposed method was implemented on top of VTM-2.0 and compared to VTM-2.0 anchor. All tests were performed on a homogeneous computer cluster of Intel Xeon E5-1620 v2 (3.7GHz) and 32GB RAM with GCC 4.7.2 compiler using CentOS Linux. In addition, the tests were conducted following the JVET common test conditions as specified in [38]. Three prediction structure configurations, “All Intra Main 10” (AI), “Random Access Main 10”, and “Low Delay B Main 10” (LDB), were tested, and four Quantization Parameters (QPs) of 22, 27, 32, and 37 were used to obtain different rate points. Table 2 lists the information on the Classes of video sequences used in Tables 3-7. For all of these tables, the runtime reduction is calculated based on the ratio of the test to the anchor. “EncT” and “DecT” refer to the total encoding time ratio and the total decoding time ratio to the anchor in the encoder and the decoder, respectively. For example, if “DecT” is less than 100%, the test reduces the decoding time compared to the anchor. Meanwhile, the BD-rate indicates the bit rate reduction ratio over the anchor achieved by the test while maintaining equivalent PSNR. For example, a negative BD-rate value means that the coding efficiency is improved. The BD-rates for the Y, Cb, and Cr components are calculated; “Overall” BD-rate means the average BD-rate for each component over all Classes, excluding Class D. In Tables 3-7, it is noted that the experiments of Class E sequences in the RA configuration and Class A1 and A2 sequences in the LDB configuration were not conducted according to the JVET common test conditions.

### TABLE 1. Comparison of computational complexity between the anchor and the proposed method for a given 8 x 8 block.

| Test                | Operation | Operations of block classification (per Luma sample) | Operations of filtering (per Luma sample) | Operations of filtering (per Chroma sample) | Total operations per sample |
|---------------------|-----------|-------------------------------------------------------|------------------------------------------|--------------------------------------------|-----------------------------|
| Anchor (VTM-2.0)    | Addition  | 28.625                                                | 25                                       | 13                                        | 79.625                      |
|                     | Comparison| 9.4375                                                | 4                                        | 1                                         | 15.4375                     |
|                     | Multiplication | 0.125                                            | 13                                       | 7                                         | 27.125                      |
|                     | Shift      | 2.5625                                                | 1                                        | 1                                         | 5.5625                      |
| Proposed method     | Addition  | 15.125                                                | 25                                       | 13                                        | 66.125                      |
| (First, second, third, and fourth patterns) | Comparison | 4.9375                                                | 4                                        | 1                                         | 10.9375                     |
|                     | Multiplication | 0.125                                            | 13                                       | 7                                         | 27.125                      |
|                     | Shift      | 1.4375                                                | 1                                        | 1                                         | 4.4375                      |
| Ratio               | Addition  | 52.84%                                                | 100%                                     | 100%                                      | 83.05%                      |
| (Proposed method / anchor) | Comparison | 52.32%                                                | 100%                                     | 100%                                      | 70.85%                      |
|                     | Multiplication | 100%                                                | 100%                                     | 100%                                      | 100%                        |
|                     | Shift      | 56.10%                                                | 100%                                     | 100%                                      | 79.78%                      |

### TABLE 2. Information on video sequences for each class.

| Class | Sequence name | Picture size | Picture count | Picture rate | Bit depth |
|-------|---------------|--------------|---------------|--------------|-----------|
| A1    | Tango2        | 3840x2160    | 294           | 60           | 10        |
|       | FoodMarket4   | 3840x2160    | 300           | 60           | 10        |
|       | Campfire      | 3840x2160    | 300           | 30           | 10        |
| A2    | CatRobot1     | 3840x2160    | 300           | 60           | 10        |
|       | DaylightRoad2 | 3840x2160    | 300           | 60           | 10        |
|       | ParkRunning3  | 3840x2160    | 300           | 50           | 10        |
| B     | MarketPlace   | 1920x1080    | 600           | 60           | 10        |
|       | RitualDance   | 1920x1080    | 600           | 60           | 10        |
|       | Cactus        | 1920x1080    | 500           | 50           | 8         |
|       | BasketballDrive | 1920x1080  | 500           | 50           | 8         |
|       | BQTerrace     | 1920x1080    | 600           | 60           | 8         |
| C     | RacerHorses   | 832x480      | 300           | 30           | 8         |
|       | BQMall        | 832x480      | 600           | 60           | 8         |
|       | PartyScene    | 832x480      | 500           | 50           | 8         |
|       | BasketballDrill | 832x480  | 500           | 50           | 8         |
| D     | RacerHorses   | 416x240      | 300           | 30           | 8         |
|       | BQSquare      | 416x240      | 600           | 60           | 8         |
|       | BlowingBubbles | 416x240    | 500           | 50           | 8         |
|       | BasketballPass | 416x240    | 500           | 50           | 8         |
| E     | FourPeople    | 1280x720     | 600           | 60           | 8         |
|       | Johnny        | 1280x720     | 600           | 60           | 8         |
|       | KristenAndSara | 1280x720 | 600           | 60           | 8         |
Table 3 summarizes the experimental results of the proposed first subsampling pattern compared to those of the anchor. For the Y component, the overall BD-rates of 0.06%, 0.07%, and 0.04% are observed for the AI, RA, and LDB configurations, respectively. In addition, the ratios of total decoding time are 97%, 97%, and 98% for the AI, RA, and LDB configurations, respectively. Furthermore, the experimental results of the proposed second subsampling pattern are summarized in Table 4. As presented in the table, the overall BD-rates are 0.03%, 0.04%, and 0.00% for the AI, RA, and LDB configurations, respectively. The total decoding time ratios of 98%, 97%, and 98% are also observed for the AI, RA, and LDB configurations, respectively. Additionally, the proposed third subsampling pattern shows the overall BD-rates of 0.03%, 0.04%, and −0.01% for the AI, RA, and LDB configurations, respectively, as listed in Table 5. The decoding times of 98%, 97%, and 98% are also observed for the AI, RA, and LDB configurations, respectively. Finally, the experimental results of the proposed fourth subsampling pattern are presented in Table 6. The overall BD-rates of 0.09%, 0.11%, and 0.06% are achieved for the AI, RA, and LDB configurations, respectively, and for the decoding time ratios, values of 97%, 97%, and 97% are observed for the AI, RA, and LDB configurations, respectively. According to the experimental results of the proposed method, the impact of the overall BD-rate is smaller...
for the sequences with bigger picture sizes, such as Classes A1 and A2, than it is for the sequences with smaller picture sizes, such as Classes B, C, and D, because adjacent samples in the pictures of the sequences of Classes A1 and A2 are more correlated than those of the sequences of Classes B, C and D. Therefore, it should be noted that this confirms the assumption that it is not necessary to consider all samples in the block to measure the local characteristics. In addition, the overall BD-rates for Classes A1 and A2 in the proposed second and third subsampling patterns are very minor and acceptable in consideration of the computational complexity reduction. Further, due to the fact that all of the proposed subsampling patterns reduce the number of the sample positions used to calculate the 1-D ML operations by half, similar decoding time reductions in the range of 2% to 3% for the proposed subsampling patterns are achieved in comparison with the anchor. Notably, the overall BD-rates are in the range of $-0.01\%$ to $0.11\%$ for all of the proposed subsampling patterns. The proposed second and third subsampling patterns show almost the same average BD-rate impacts, as the subsampled sample positions of both patterns are very similar to each other. These two patterns also show relatively small coding losses compared to the proposed first and fourth subsampling patterns. The coding loss of the proposed fourth subsampling pattern is relatively high, but this pattern can be efficiently implemented by SIMD optimization to further reduce the computational complexity. The experimental results show that the proposed second and third subsampling patterns based on the quincunx pattern achieve reasonable results in terms of both the decoding runtime reduction and the coding loss as compared to the proposed first and fourth subsampling patterns.

In addition, to compare the proposed method with another subsampling method in terms of the tradeoff between the decoding time and the BD-rate impact, a SSML operator using a subsampling factor of 2 that calculates 1-D ML operations at every second sample position in both the horizontal and vertical directions within an $8 \times 8$ window was tested against the anchor. In this method, 1-D ML operation is calculated for the top-left sample position only in every $2 \times 2$ region within the $8 \times 8$ window. Aside from the window size, summation operation over the window, and different number of gradient directions, this method uses the same sample positions proposed in [29], [30] for HEVC. This method is also the same as the subsampled gradient calculation method proposed in [39]–[41] for VVC aside from the fact that the subsampled gradient calculation method only uses the subsampled sample positions for the highest temporal layer, where the worst-case operation counts cannot be reduced. However, the SSML operator using the subsampling factor of 2 can reduce the worst-case operation counts by applying the subsampled sample positions irrespective of the temporal layer. Table 7 presents the experimental results of the SSML operator using the subsampling factor of 2.

As listed in Table 7, compared to all of the proposed subsampling patterns, the SSML operator using the subsampling factor of 2.

### TABLE 6. Experimental results of the proposed fourth subsampling pattern.

| Sequence | All Intra Main 10 | Random Access Main 10 | Low Delay B Main 10 |
|----------|------------------|----------------------|-------------------|
|          | Y    | Cb   | Cr   | EncT | DecT | Y    | Cb   | Cr   | EncT | DecT | Y    | Cb   | Cr   | EncT | DecT |
| Class A1 | 0.07% | 0.00% | 0.00% | 100% | 97%  | 0.04% | 0.00% | -0.04% | 100% | 96%  | ---  | ---  | ---  | ---  | ---  |
| Class A2 | 0.05% | 0.00% | 0.00% | 100% | 97%  | 0.13% | 0.05% | 0.01%  | 100% | 97%  | 0.11% | -0.04% | -0.15% | 100% | 97%  |
| Class B  | 0.08% | -0.01% | -0.01% | 100% | 97%  | 0.14% | 0.05% | -0.12% | 100% | 97%  | 0.08% | 0.00% | -0.07% | 100% | 98%  |
| Class C  | 0.10% | -0.02% | -0.02% | 100% | 98%  | 0.11% | 0.05% | -0.11% | 100% | 97%  | 0.08% | -0.03% | -0.17% | 100% | 97%  |
| Class E  | 0.17% | -0.02% | -0.02% | 100% | 97%  | ---  | ---  | ---  | ---  | ---  | 0.05% | -0.07% | -0.31% | 100% | 97%  |
| Overall  | 0.09% | -0.01% | -0.01% | 100% | 97%  | 0.11% | 0.04% | -0.07% | 100% | 97%  | 0.06% | -0.03% | -0.17% | 100% | 97%  |
| Class D  | 0.07% | -0.02% | -0.02% | 100% | 98%  | 0.04% | -0.02% | -0.14% | 100% | 98%  | 0.08% | 0.66% | -0.21% | 100% | 98%  |

### TABLE 7. Experimental results of SSML operator using subsampling factor of 2.

| Sequence | All Intra Main 10 | Random Access Main 10 | Low Delay B Main 10 |
|----------|------------------|----------------------|-------------------|
|          | Y    | Cb   | Cr   | EncT | DecT | Y    | Cb   | Cr   | EncT | DecT | Y    | Cb   | Cr   | EncT | DecT |
| Class A1 | 0.13% | -0.01% | -0.01% | 99%  | 96%  | 0.13% | 0.12% | -0.04% | 99%  | 95%  | ---  | ---  | ---  | ---  | ---  |
| Class A2 | 0.09% | 0.00% | 0.00% | 99%  | 96%  | 0.20% | 0.00% | -0.05% | 99%  | 96%  | 0.17% | -0.03% | -0.39% | 99%  | 96%  |
| Class B  | 0.16% | -0.01% | -0.01% | 100% | 96%  | 0.28% | 0.02% | 0.01%  | 99%  | 96%  | 0.18% | -0.10% | 0.05%  | 100% | 96%  |
| Class C  | 0.25% | -0.03% | -0.03% | 100% | 97%  | 0.22% | 0.07% | 0.00%  | 100% | 97%  | 0.17% | -0.40% | -0.16% | 99%  | 96%  |
| Class E  | 0.41% | -0.05% | -0.05% | 100% | 96%  | ---  | ---  | ---  | ---  | ---  | 0.22% | 0.05% | -0.02% | 99%  | 96%  |
| Overall  | 0.20% | -0.02% | -0.02% | 100% | 96%  | 0.22% | 0.05% | -0.02% | 99%  | 96%  | 0.17% | -0.15% | -0.18% | 99%  | 96%  |
| Class D  | 0.17% | -0.05% | -0.05% | 100% | 97%  | 0.17% | 0.03% | -0.13% | 100% | 97%  | 0.16% | 0.35% | -0.12% | 99%  | 97%  |
tor of 2 shows a relatively larger coding loss while reducing the total decoding time ratios by an additional 1% to 2%. Further, compared to the proposed second and third subsampling patterns, the coding losses in the overall BD-rates for the AI, RA, and LDB configurations are more than seven times larger than those of the proposed second subsampling pattern. Therefore, it should be noted that all of the proposed subsampling patterns, particularly for the proposed second and third subsampling patterns, show a better tradeoff between the decoding time and BD-rate impact.

V. CONCLUSION

In this paper, four different subsampling patterns based on the 4 × 4 block classification for SSML operator were proposed to reduce the computational complexity of the ALF. All of the proposed subsampling patterns calculate the 1-D ML values on the subsampled sample positions within the 8 × 8 window, so the number of sample positions used to calculate the 1-D ML operation is reduced by half. The experimental results demonstrate that the proposed method reduced the total decoding time in the range of 2% to 3% under negligible BD-rate impacts compared to VTM-2.0. It is also worth noting that the second subsampling pattern of the proposed SSML operator was adopted into WD 3 and VTM-3.0 reference software, and included in FDIS of VVC.

REFERENCES

[1] B. Bross, J. Chen, S. Liu, and Y.-K. Wang, Versatile Video Coding (Draft 10), document JVET-S2001, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/ WG 11, 19th JVET Meeting, Jul. 2020.

[2] Advanced Video Coding (AVC), Standard ITU-T Recommendation H.264 and ISO/IEC 14496-10, May 2003.

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 560–576, Jul. 2003.

[4] High Efficient Video Coding (HEVC), Standard ITU-T Recommendation H.265 and ISO/IEC 23008-2, Apr. 2013.

[5] G. Bjontegaard, “Calculation of average PSNR Differences Between RD Curves,” document VCEG-M33, ITU-T SG 16 Q 6 Video Coding Experts Group (VCEG), 13th VCEG Meeting, Apr. 2001.

[6] G. Bjontegaard, “Improvements of the BD-PSNR Model,” document VCEG-AH4, 13th VCEG Meeting, Jul. 2008.

[7] I. S. Chong, M. Karczewicz, C.-Y. Chen, C.-M. Fu, C.-Y. Tsai, Y.-W. Huang, S. Lei, T. Yamakage, T. Chujoh, and T. Watanabe, 3rd JVET Meeting, May 2016.

[8] J. Chen, Y. Ye, and S. Kim, Algorithm description for Versatile Video Coding and Test Model 10 (VTM 10), document JVET-S2002, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 19th JVET Meeting, Jul. 2020.

[9] W.-J. Chien, J. Boyce, W. Chen, Y.-W. Chen, R. Chernyak, K. Choi, R. Hashimoto, Y.-W. Huang, H. Jang, R.-L. Liao, and S. Liu, JVET AHG Report: Tool Reporting Procedure (AHG13), document JVET-S0013, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 19th JVET Meeting, Jul. 2020.

[10] M. Karczewicz, L. Zhang, W.-J. Chien, and X. Li, EE2.5: Improvements on Adaptive Loop Filter, document JVET-B0060, Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 2nd JVET Meeting, Feb. 2016.

[11] M. Karczewicz, L. Zhang, W.-J. Chien, and X. Li, EE2.5: Improvements on Adaptive Loop Filter, document JVET-C0038, Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 3rd JVET Meeting, May 2016.
SUNG-CHANG LIM received the B.S. (Hons.) and M.S. degrees in computer engineering from Sejong University, Seoul, South Korea, in 2006 and 2008, respectively, where he is currently pursuing the Ph.D. degree. Since 2005, he has been involved with the development of various video coding standards, including AVC FRExt and MVC Extension, HEVC, and VVC with more than 50 technical contributions. In 2008, he joined the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, where he is a Senior Researcher with the Communication and Media Research Laboratory. He has authored or coauthored over 50 publications. He holds numerous internationally issued patents and patent applications in video coding. His research interests include video coding and image processing.

MYUNGJUN KIM received the B.S. and M.S. degrees in computer engineering from Sejong University, South Korea, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree. His research interests include video compression, image processing, machine learning for video coding, and CNN-based video coding.

JUNGWON KANG (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Korea Aerospace University, Seoul, South Korea, in 1993 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2003. Since 2003, she has been a Principal Member (Research Staff) with the Communication and Media Research Laboratory, Electronics and Telecommunications Research Institute, South Korea. She has made contributions with HEVC and VVC standards. Her current research interests include video/image compression, immersive video processing, and multimedia applications.

YUNG-LYUL LEE (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Sogang University, Seoul, South Korea, in 1983 and 1987, respectively, and the Ph.D. degree in electrical and electronic engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1999. He was a Principal Researcher with Samsung Electronics Research and Development Center from 1987 to 2001. He has been a Professor with the Department of Computer Engineering, Sejong University, Seoul, since 2001. He was a Visiting Scholar with The University of Texas at Arlington, USA, from September 2006 to August 2007. He is currently the Vice-Chairman with the Korea Institute of Broadcasting and Media Engineering. He had contribution documents to AVC/H.264, high efficiency video coding (HEVC), and versatile video coding (VVC) standards. He holds patents in AVC/H.264 and HEVC standard. His current research interests include video compression, image processing, 3-D video coding, deep learning, and CNN-based video coding. He received the Minister Prize from the Ministry of Commerce, Industry and Energy, South Korea, in November 2006, and the Korea Science Technology Superiority Paper Prize in October 2006.