# Overview of the Screen Content Support in VVC: Applications, Coding Tools, and Performance

Tung Nguyen, Xiaozhong Xu, *Member, IEEE*, Felix Henry, Ru-Ling Liao, Mohammed Golam Sarwer, Marta Karczewicz, Yung-Hsuan Chao, Jizheng Xu, *Senior Member, IEEE*, Shan Liu, *Senior Member, IEEE*, Detlev Marpe, *Fellow, IEEE*, and Gary J. Sullivan, *Fellow, IEEE*

*(Invited Paper)*

***Abstract*— In an increasingly connected world, consumer video experiences have diversified away from traditional broadcast video into new applications with increased use of non-camera-captured content such as computer screen desktop recordings or animations created by computer rendering, collectively referred to as screen content. There has also been increased use of graphics and character content that is rendered and mixed or overlaid together with camera-generated content. The emerging Versatile Video Coding (VVC) standard, in its first version, addresses this market change by the specification of low-level coding tools suitable for screen content. This is in contrast to its predecessor, the High Efficiency Video Coding (HEVC) standard, where highly efficient screen content support is only available in extension profiles of its version 4. This paper describes the screen content support and the five main low-level screen content coding tools in VVC: transform skip residual coding (TSRC), block-based differential pulse-code modulation (BDPCM), intra block copy (IBC), adaptive color transform (ACT), and the palette mode. The specification of these coding tools in the first version of VVC enables the VVC reference software implementation (VTM) to achieve average bit-rate savings of about 41% to 61% relative to the HEVC test model (HM) reference software implementation using the Main 10 profile for 4:2:0 screen content test sequences. Compared to the HM using the Screen-Extended Main 10 profile and the same 4:2:0 test sequences, the VTM provides about 19% to 25% bit-rate savings. The same comparison with 4:4:4 test sequences revealed bit-rate savings of about 13% to 27% for $Y'C_BC_R$ and of about 6% to 14% for $R'G'B'$ screen content. Relative to the HM without the HEVC version 4 screen content coding extensions, the bit-rate savings for 4:4:4 test sequences are about 33% to 64% for $Y'C_BC_R$ and 43% to 66% for $R'G'B'$ screen content.**

***Index Terms*— VVC, H.266, video coding, screen content coding, TSRC, BDPCM, IBC, palette coding, ACT, HEVC, VTM.**

## I. Introduction

The Versatile Video Coding (VVC) standard (Rec. ITU-T H.266 and ISO/IEC 23090-3) [1]–[3] includes low-level coding tools specifically developed for screen content signals, highlighting the importance of screen content videos that have become increasingly prominent since the specification of the first version of the High Efficiency Video Coding (HEVC) in 2013 [4], [5]. Screen content is an umbrella term for non-camera captured signals, and typically refers to computer-generated text, graphics and animations. Due to its different signal characteristics, such as sharp edges, repeated patterns, or the non-existence of a noise level, not all coding technologies suitable for camera-captured content are efficient for screen content. A further aspect showing a divergence relative to camera-captured signals is subjective quality perception, where coding tools optimized for camera-captured content can tend to cause blurriness, ringing, and "mosquito" artifacts with computer-generated content in a way that is commonly more distracting to viewers.

The popularity of screen content is partially due to the progress during the last decade that has shown a significant shift in how users experience media, especially video. For example, the communication channels have shifted from traditional broadcast towards network-based services, new kinds of viewing devices such as smartphones have appeared, and completely new applications have emerged, such as online education or live streaming of computer games. Moreover, there has been increased use of graphics and character content that is rendered and mixed or overlaid together with camera-generated content. VVC includes the following low-level screen content coding tools already in its first version to address the developments above:

- Transform skip residual coding (TSRC) [6]
- Block-based differential pulse-code modulation (BDPCM) [7]
- Intra block copy (IBC) [8]

Manuscript received August 22, 2020; revised February 15, 2021; accepted March 6, 2021. Date of publication April 20, 2021; date of current version October 4, 2021. This article was recommended by Associate Editor W. Zhu. *(Corresponding author: Tung Nguyen.)*

Tung Nguyen and Detlev Marpe are with the Department of Video Communication and Applications, Fraunhofer Institute for Telecommunications–Heinrich Hertz Institute, 10587 Berlin, Germany (e-mail: tung.nguyen@hhi.fraunhofer.de; detlev.marpe@hhi.fraunhofer.de).

Xiaozhong Xu and Shan Liu are with Tencent, Palo Alto, CA 94036 USA (e-mail: xiaozhongxu@tencent.com; shanl@tencent.com).

Felix Henry is with Orange Labs, 92794 Issy-les-Moulineaux, France (e-mail: felix.henry@orange.com).

Ru-Ling Liao and Mohammed Golam Sarwer are with Alibaba, Sunnyvale, CA 94085 USA (e-mail: ruling.lrl@alibaba-inc.com; m.sarwer@alibaba-inc.com).

Marta Karczewicz and Yung-Hsuan Chao are with Qualcomm Technologies, Inc., San Diego, CA 92121 USA (e-mail: martak@qti.qualcomm.com; yunghsua@qti.qualcomm.com).

Jizheng Xu is with Bytedance Inc., San Diego, CA 92122 USA (e-mail: xujizheng@bytedance.com).

Gary J. Sullivan is with Microsoft, Redmond, WA 98052 USA (e-mail: garysull@microsoft.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2021.3074312.

Digital Object Identifier 10.1109/TCSVT.2021.3074312

- Adaptive color transform (ACT) [9]
- Palette mode [10]

Both TSRC and BDPCM are coding tools that did not exist in an equivalent form in HEVC, and they are integrated with the existing transform skip mode (TSM) that was specified in version 1 of HEVC [11]. IBC, ACT, and the palette mode are coding tools inherited from the HEVC screen content coding (SCC) extensions [12], [13] with design refinements to balance between coding performance and implementation feasibility. This paper gives an overview of the screen content coding support in VVC and a description of the above mentioned low-level coding tools, including:

- The history of development of specific coding tools for screen content, starting in HEVC and up to the final design in the first VVC version,
- The basic screen content support in VVC and the profiles that specify its associated low-level coding tools,
- The two coding tools in VVC that extend the operational range of the TSM for screen content: TSRC and BDPCM,
- The modifications to the HEVC low-level coding tools IBC, ACT, and the palette mode to suit the VVC architecture, and
- The compression efficiency of VVC for screen content applications relative to the first HEVC version and its SCC extensions.

For further information about the VVC standard and the history of its development, the reader is referred to [2], [3]. For an explanation of the principles of its high-level syntax, such as its use of parameter sets and slices, see [14].

The organization of the paper is as follows. Section II briefly describes the development process of screen content started in HEVC to the current design in VVC. The following Section III describes the two coding tools that are incorporated in the TSM coding path. Section IV describes modifications to the three coding tools inherited from the HEVC version 4 SCC extensions. Experimental results showing the performance of the screen content coding tools are presented in Section V, and conclusions are provided in Section VI.

## II. Screen Content Support in HEVC and VVC

The last decade showed a technology and market demand for higher compression efficiency of screen content due to the increased prevalence of such content in many mainstream applications. When this strong demand for improved screen content coding appeared, which was after the finalization of the first HEVC version, ISO/IEC JTC 1/SC 29/WG 11 (MPEG) and ITU-T SG16/Q6 (VCEG) mandated the Joint Collaborative Team on Video Coding (JCT-VC) responsible for the HEVC standardization activity to develop HEVC extension profiles with a focus on higher compression efficiency for screen content. The HEVC version 4 extensions comprise the SCC extensions [15] that extended the HEVC capability for screen content applications by incorporating four main low-level coding tools: IBC, ACT, palette mode, and adaptive motion vector resolution (AMVR) [16]. However, services have to use software solutions that are typically less cost-effective than hardware implementations for screen content applications. The reason behind that is the cost of developing hardware for the newer HEVC versions, also hindering the adoption of the HEVC SCC extensions for end-consumer applications. As VVC includes low-level coding tools designed for screen content signals in its first version, one may expect a more widespread usage of VVC in screen content applications due to hardware availability and deployment.

A key issue for screen content is the chroma resolution. For consumer video applications that emphasize camera-captured content, the typical format is what is known as 4:2:0, where the video consists of a luma plane that represents sampled brightness and two chroma planes that each have half the width and half the height of the luma plane. With screen content, the presence of sharp edges can benefit from use of the 4:4:4 format, for which the three planes all have the same resolution. A less common format historically used in studio environments is 4:2:2, in which the chroma planes have the same height and half the width of the luma plane. Additionally, "monochrome" video with only one plane of samples can also be an important format, e.g., for carrying alpha transparency maps for video overlays or depth maps for 3D video applications. In video coding specifications, chroma format support is often a part of the specification of feature profiles, such that lower capability profiles support only 4:2:0 (and perhaps monochrome) video and higher capability profiles additionally support 4:4:4 and 4:2:2.

In terms of complexity and application importance, a best-effort support for screen content appeared with the first HEVC version in the form of the TSM. From an encoder point-of-view, the TSM encoding process completely bypasses the transform stage, which is useful since screen content residual signals typically show no or negligible energy compaction after applying a transform. Due to the usage of screen content signals in niche markets at that time, the specification permits TSM for 4 × 4 transform blocks only. HEVC version 2 introduced the Range Extensions (RExt) [17] specifying higher bit depths and non-4:2:0 chroma formats. The extension of TSM to transform blocks with size of up to 32 × 32 further extended the support for screen content in HEVC RExt. Although not specifically designed for screen content, residual differential pulse code modulation (RDPCM) [18], cross-component prediction (CCP) [19], and other changes such as the initialization of the Rice parameter for the binarization in the entropy coder, the rotation of the TSM levels by 180°, and using a single context model for coding the significance map further improve the compression efficiency for screen content. Dedicated screen content support finally came with the Screen-Extended profiles in HEVC version 4 and included a set of low-level screen content coding tools in 2016. In contrast to the motivation for adoption of IBC, ACT, and the palette mode, the Joint Video Experts Team (JVET) responsible for the VVC development initially adopted AMVR into the VVC standard due to its performance for camera-captured content rather than specifically for screen content. For this reason, this paper does not cover the AMVR coding tool in VVC, although VVC does include some refinement of AMVR relative to HEVC (in particular, the motion vector resolution in VVC is selectable locally for each coding unit rather than being a whole-slice property). For further information on AMVR and other aspects of motion vector coding in VVC, the reader is referred to [20].

HEVC version 4 also extended the existing profiles by additional Screen-Extended profiles that include the introduced low-level screen content coding tools. Specifically, all coding tools except ACT are available for the 4:2:0 and the 4:4:4 chroma formats, whereas ACT is only available for the 4:4:4 chroma format as $R'G'B'$ content are usually without chroma subsampling. The VVC development started without dedicated screen content coding tools except for TSM, which was initially limited to 4 × 4 transform blocks as in HEVC version 1. The JVET adopted all the low-level screen content coding tools of the HEVC SCC extensions after several refinements and optimization cycles into the VVC draft for efficient screen content support. Besides the known coding tools from HEVC, the JVET adopted with TSRC and BDPCM coding tools that did not exist in HEVC to further improve screen content representation. VVC version 1 specifies three profiles that support only the 4:2:0 and monochrome chroma formats (the Main 10, Multilayer Main 10 and Main 10 Still Picture profiles) and three corresponding profiles that additionally support the 4:4:4 and 4:2:2 chroma formats (the Main 10 4:4:4, Multilayer Main 10 4:4:4 and Main 10 4:4:4 Still Picture profiles). In each of these groups of three profiles, there is a profile for ordinary single-layer video coding, a profile supporting multi-layer features such as scalable video coding, and a profile for still-picture coding. In contrast to the way coding tools are allocated to profiles in the HEVC SCC extensions, the palette mode is available only in the 4:4:4 profiles of VVC, whereas the TSM coding tools (extension of TSM, TSRC, and BDPCM) and IBC are also available in the 4:2:0 profiles. Note that due to the "onion-shell" design of the profile features, the palette mode can still be applied for the 4:2:0 chroma format in VVC, but only in the 4:4:4 profiles (i.e., using a 4:4:4 profile for coding a 4:2:0 video sequence).

## III. Transform Skip Coding Tools

HEVC version 1 supports screen content in a best-effort manner with the specification of the TSM, but limits its use to 4 × 4 transform blocks. The HEVC RExt profiles remove the restriction to 4 × 4 transform blocks, allowing the usage of TSM for all transform block sizes, i.e., up to 32 × 32, which is also the maximum allowed transform block size of the HEVC specification. In RExt, a conforming encoder signals the maximum allowed TSM transform block size ($\text{size}_{TSM}^{max}$) in the picture parameter set (PPS) as

$$\text{TSM}_{max} = \log_2(\text{size}_{TSM}^{max}) - 2. \tag{1}$$

In the HEVC RExt and the HEVC SCC Common Test Conditions (CTC) [21], [22], the default maximum block size for TSM is 4 × 4 ($\text{TSM}_{max} = 0$) as there was no efficient implementation for a fast encoding operation mode for variable block size TSM at that time in the reference software used for the CTC.

The VVC development started with the maximum allowed transform block size equal to 4 × 4 for TSM, the same configuration as in HEVC version 1. The JVET adopted an extension to transform block sizes of up to 32 × 32 for TSM using the same high-level syntax as in the HEVC RExt specification during the development of VVC after it became clear that a fast encoder control is feasible [23]. Without loss of generality, this paper covers rectangular shapes supported in VVC by the term block, i.e., a shape fulfills the condition for TSM when its width $W$ and its height $H$ are less than or equal to the maximum allowed block size, i.e.,

$$W \leq \text{size}_{TSM}^{max} \wedge H \leq \text{size}_{TSM}^{max}. \tag{2}$$

To this end, VVC employs a second residual coding stage optimized for non-transformed quantized indices, referred to as TSRC. While TSRC extends the end of the TSM pipeline, BDPCM extends the TSM pipeline's beginning by an intra prediction mode that exclusively uses the TSM path.

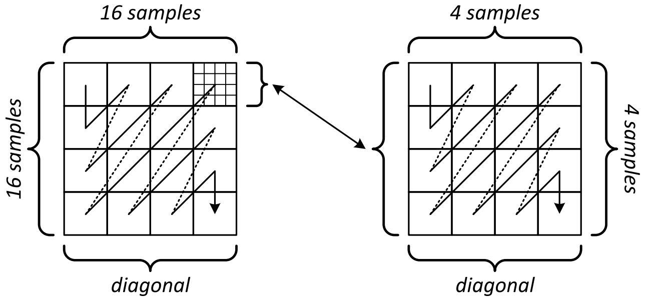


Fig. 1. The illustration shows exemplarily the partitioning into smaller 4 × 4 SBs and the corresponding processing order given by the diagonal scanning pattern for a 16 × 16 block. Note that the processing is SB-wise, i.e., the diagonal scanning pattern over the whole block is 4 × 4 sub-sampled (on the left), whereas the scanning pattern for sample positions covered by the SBs is the same as for a regular 4 × 4 transform block (on the right).

### A. Transform Skip Residual Coding (TSRC)

The main residual coding stage in VVC, informally referred to as regular residual coding (RRC) [24] in the following paragraphs, served as the starting point for the TSRC design. Both TSRC and RRC build upon the concept of 4 × 4 sub-blocks (SBs) introduced in HEVC for variable transform block sizes [25] and often also referred to as coefficient groups (CGs) [24]. Fig. 1 illustrates the decomposition of a block larger than 4 × 4 into 4 × 4 SBs and the associated diagonal scanning pattern for TSRC. Since the TSRC design is derived from the RRC design, the following paragraph gives a brief overview of the RRC design followed by the modifications resulting in the TSRC design. Note that VVC supports non-4 × 4 sub-shapes due to the support of non-square rectangular shapes for transform blocks, which contrasts with HEVC supporting 4 × 4 SBs only. However, with no loss of generality, the following description uses the term SB to also include non-4 × 4 sub-shapes within a transform block for the parsing process.

The RRC design uses a multi-stage indication of significance, i.e., existence of non-zero levels embedded into the 4 × 4 SB concept [26] in order to provide an efficient signaling for the transformed and quantized residual signals. Before processing the transform coefficient levels of a transform block, the syntax element $s_{cbf}$ appears in the bitstream syntax, indicating the significance for the whole transform block with a $s_{cbf} = 0$ implying that all levels within the transform block are equal to zero. When $s_{cbf} = 1$ indicating the existence of at least one absolute level that is not equal to zero, then the processing continues with the last significant scan position using the variables $\mathrm{v}_{last}^{X}$ and $\mathrm{v}_{last}^{Y}$. As their names indicate, the variables specify the position of the last significant transform coefficient within the transform block in terms of column and row position, respectively, each relative to the transform block's top-left corner when scanning along the diagonal scanning pattern as shown in Fig. 1. This means that when scanning beyond this last significant scan position along the diagonal scanning pattern towards the end of the block, there are no further significant transform coefficient levels in the block. Note that the value of each of the two variables $\mathrm{v}_{last}^{X}$ and $\mathrm{v}_{last}^{Y}$ is reconstructed from two syntax elements specifying a prefix and (possibly empty) suffix part of the column or row position of the last significant position (for more details see [24]).

The level parsing process in RRC starts at the last significant scan position and uses the reverse diagonal scanning pattern to process all frequency positions that lie on the scanning path towards the DC frequency position. For each $4 \times 4$ SB of the transform block, the syntax element $s_{csb}$ indicates the existence of significant levels within the current SB. There are two exceptions, the first for the SB covering the last significant scan position and the second for the SB covering the DC frequency position, where a conforming decoder infers $s_{csb} = 1$. Since the last significant scan position already specifies a significant level within the SB, the decoder can infer $s_{csb} = 1$ for the corresponding SB. Due to the high likelihood of transmitting a significant level for the SB covering the DC frequency position, because of the energy compaction property of the transform, a conforming decoder always infers $s_{csb} = 1$ for the top-left SB. However, there is no inference rule for the syntax element $s_{sig}$ (specifying the significance of a scan position) of the top-left SB as for the other SBs, i.e., when no significant level occurs within the first 15 scan positions, the SB's last scan position has to be significant due to $s_{csb} = 1$. In the case of a $4 \times 4$ transform block, the decoder infers $s_{csb}$ to be equal to 1 as the one and only SB fulfills both special conditions described above.

*1) Processing Order:* A major difference between RRC and TSRC is the reverse order of processing the scanning pattern, i.e., the usage of the regular forward scan order instead of the reverse scan order along the diagonal scanning pattern for TSRC, as illustrated in Fig. 1. That is because usually intra prediction becomes less efficient for sample positions further away from the employed reference samples, located above and to the left of the current block. This also implies that the local residual signal variance in the spatial domain becomes larger with increasing distance to the reference samples, resulting in larger level values at the right-bottom corner of the block. By reversing the reverse scan order, i.e., by using the regular forward diagonal scan order, the probability for significant levels increases in scan order, similar to the case in RRC, when operating along the reverse scan order in the frequency domain. Although the benefit for inter-predicted blocks is not significant, changing the scanning pattern direction does not adversely affect the performance in such cases. Note that the residual rotation by 180° as used in HEVC RExt exploits the same phenomena, but it is effectively not the same as changing the scanning direction due to the SB-wise processing.

The second major difference is the elimination of signaling the last significant scan position in TSRC, resulting in the processing of all scan positions within the block, as illustrated in Fig. 1 as well. However, the multi-stage significance signaling approach remains valid in the sense that $s_{csb}$ appears for all SBs of the transform block, where the last (bottom-right) SB of the transform block constitutes an exception. For that special SB, a conforming decoder can infer $s_{csb} = 1$, but only when $s_{csb} = 0$ for all previously processed SBs.

*2) Level Coding:* As in the RRC design [24], the coding and decoding of quantization indices (levels) for TSRC proceeds in several coding passes within an SB, i.e., the parsing process, from a decoder point-of-view, iterates multiple times over each scan position along the scanning pattern within an SB until the decoder can reconstruct the full level information. For TSRC, the levels are coded (and decoded) in three passes, provided that the limit on context-coded bins is not exceeded (see subsection III-A.4 below). The first coding pass includes up to 4 context-coded syntax elements for each level in an interleaved fashion: $s_{sig}$, $s_{sign}$, $s_{gt1}$, and $s_{par}$, indicating the significance, the sign, the greater-than-1 flag as well as the parity flag, respectively. It is worth to note that the coding of $s_{sign}$ in RRC uses the bypass mode of the CABAC engine and appears as a final pass, whereas the coding of $s_{sign}$ in TSRC is part of the first coding pass that employs adaptive context models. While the overall probability of $s_{sign}$ remains roughly equal to 0.5 in TSRC, there is often a bias towards a certain direction locally for transform skipped residual signals in the spatial domain. Using adaptive context models exploiting that statistical anomaly improves the coding efficiency for TSRC. Note that the first pass also includes the $s_{par}$ syntax element (specifying the parity of the level) that was introduced for trellis-coded quantization (TCQ) [27]. Although TCQ is not active for TSRC, the binarization process for TSRC keeps $s_{par}$ in order to maintain a similar list of syntax elements for the first coding pass of both residual coding paths.

The second coding pass in TSRC consists of up to 4 additional bins that are context-coded indicating for each absolute level its greater-than-x ($gtx$) property relative to the threshold values $x \in \{3, 5, 7, 9\}$ with the associated syntax elements $s_{gt3}$, $s_{gt5}$, $s_{gt7}$, and $s_{gt9}$. The third coding pass finally specifies the remainder of those absolute levels that were not fully transmitted in the preceding coding passes, and all such related $s_{rem}$ syntax elements are binarized by a concatenation of truncated Rice (TR) and Exp-Golomb (EG) bin strings with each bin bypass-coded as in the RRC path.

Conceptually, there exists a direct relationship between the binarization and the absolute level coding process. Fig. 2 illustrates the binarization process in HEVC and VVC, consisting of the concatenation of 3 different prefix codes with varying parameterization. As a matter of principle, all bins related to the truncated unary (TU) bin string are context-coded using adaptive context models, whereas the bins related to the TR and EG bin strings are coded using the bypass mode of the CABAC engine. The most prominent difference

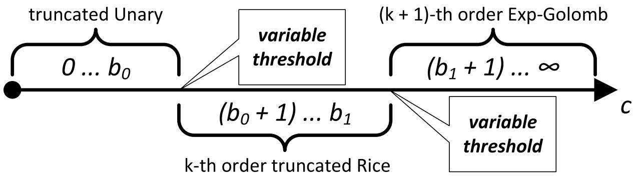


Fig. 2. The illustration summarizes the binarization process employed in both HEVC and VVC, which concatenates three different prefix codes. However, the parameterization for the binarization process differs between VVC and HEVC. While the second variable bound depends on the selected Rice parameter, the first variable bound depends on the remaining budget for context-coded bins.

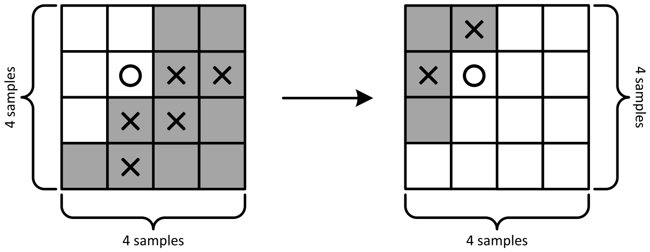


Fig. 3. The illustration shows the local template for context modeling in RRC (on the left) using five neighboring frequency positions (marked with an ×) and the local template for TSRC (on the right) using two neighboring sample positions in the spatial domain (marked with an ×). White-shaded box shapes denote frequency or sample positions not processed so far, whereas gray-shaded box shapes denote already processed frequency or sample positions. The current scan position to be processed is marked with a circle.

between the HEVC and VVC binarization process is the parity $s_{par}$ syntax element for TCQ [27], For RRC, when TCQ is enabled, the parity flag drives the TCQ state machine, and its signaling in the first coding pass avoids the requirement for reconstruction of the full level information for each scan position before knowing the TCQ state of the next scan position. Note that the TCQ state is an important element in context modeling of the $s_{sig}$ bin in RRC [24]. From a binarization perspective, each pair of two consecutive absolute levels $\{(2n, 2n+1) | n \in \mathbb{N}^{+}\}$ with different parity shares the same binary codeword representation. As a consequence, the $c$-axis representing the absolute level values in Fig. 2 has to be subsampled by a factor of two for all $c \geq 2$ resulting in an application of the binarization process to the set of absolute levels $c \in \{0, 1\} \cup \{2n | n \in \mathbb{N}^{+}\}$ instead of $c \in \mathbb{N}_0$ for HEVC. Another consequence is that the resulting bin strings need to be completed by adding the parity bin representing $s_{par}$ at the appropriate bin index whenever $c \geq 2$. In both RRC and TSRC, the index order of the associated TU bin strings is such that the parity bin is always placed after the greater-than-1 bin representing $s_{gt1}$. Note that for TSRC a further bin representing the syntax element $s_{sign}$ has to be added right after the $s_{sig}$ bin for all $c \geq 1$, as already discussed above. For the remainder coding pass, TSRC employs the same second threshold $b_1$ for TR and EG binarization and a fixed Rice parameter $k = 1$.

*3) Context Modeling:* For context modeling, TSRC employs a local template $\boldsymbol{T}$ similar to the RRC path but with fewer neighboring positions for the evaluation. Fig. 3 illustrates the local template and the already processed neighboring frequency positions for the RRC path on the left and the corresponding local template for TSRC on the right. Note that the local template spans to the right and the bottom in RRC because of the reverse diagonal scanning pattern, whereas in TSRC, the local template is inverted, i.e., it spans to the left and the top of the current scan position in the spatial domain. Instead of evaluating five neighbors, the context modeling in TSRC only evaluates the spatial neighbors above and to the left of the current scan position.

Let $\chi \in \boldsymbol{X}$ denote the context memory offset within a context model set $\boldsymbol{X}$, where each context model set $\boldsymbol{X}$ has a fixed offset so that all context model sets are disjoint. Furthermore, let $\boldsymbol{T}_A(\mathrm{s})$ and $\boldsymbol{T}_L(\mathrm{s})$ be the evaluation outcomes of the above neighbor and the left neighbor, respectively. Then, the context model index $\chi_{sig} \in \boldsymbol{X}_{sig}$ for the syntax element $s_{sig}$ with $|\boldsymbol{X}_{sig}| = 3$ is equal to

$$\chi_{sig} = \boldsymbol{T}_A(s_{sig}) + \boldsymbol{T}_L(s_{sig}). \tag{3}$$

The same rule applies for $s_{gt1}$ with the exception for the BDPCM mode that triggers a single dedicated context model ($|\boldsymbol{X}_{gt1}| = 4$):

$$\chi_{gt1} = \begin{cases} \boldsymbol{T}_A(s_{sig}) + \boldsymbol{T}_L(s_{sig}) & \text{BDPCM} = 0 \\ 3 & \text{BDPCM} = 1. \end{cases} \tag{4}$$

The context modeling for the $s_{sign}$ syntax element compares the neighboring sample values covered by the local template and selects the first context model ($\chi_{sign} = 0$) of the sign context model set $\boldsymbol{X}_{sign}$ with $|\boldsymbol{X}_{sign}| = 6$ when both neighbors are insignificant ($\boldsymbol{T}_A(s_{sig}) = 0 \wedge \boldsymbol{T}_L(s_{sig}) = 0$) or the two sign values differ ($\boldsymbol{T}_A(s_{sign}) \neq \boldsymbol{T}_L(s_{sign})$). Note that the second condition only holds if both neighboring positions consist of significant levels. When one out of the two neighboring positions consists of a positive level ($\boldsymbol{T}_A(s_{sign}) = 1 \vee \boldsymbol{T}_L(s_{sign}) = 1$), the context modeling selects the set's second context model ($\chi_{sign} = 1$). All other cases lead to the usage of the third context model ($\chi_{sign} = 2$). Note that for the BDPCM mode, the same context modeling for the $s_{sign}$ syntax element applies, but with a fixed offset equal to 3, which means that for BDPCM a separate set of context models is used. The $s_{par}$ syntax element which belongs to the first coding pass and all syntax elements of the second coding pass, i.e., $s_{gt3}$, $s_{gt5}$, $s_{gt7}$, and $s_{gt9}$, use a single dedicated context model.

*4) Limit on Context-Coded Bins:* For enabling cost-effective and feasible hardware implementations a worst-case limit of context-coded bins ($ccb$) per coefficient in a transform block was incorporated into the design for both RRC and TSRC. When $N$ denotes the number of transform coefficients or samples in a (transform) block, the maximum budget of context-coded bins is set to $\boldsymbol{B}_{ccb} = 1.75 \times N$ at the beginning of the level coding. For each context-coded coding pass and each scan position, the condition $\boldsymbol{B}_{ccb} \geq 4$ has to be checked before the actual execution of the coding pass. Whenever $\boldsymbol{B}_{ccb} < 4$ occurs for a scan position, the processing of remaining level information falls back to a pure bypass coding mode by using the binarization consisting of TR and EG bin strings only, i.e., the first variable threshold $b_0$ shown in Fig. 2 is set equal to the value of $-1$.

*5) Level Prediction:* The processing applies a level prediction technique at the end of the remainder coding pass when the current scan position has level information coded in the first coding pass. From the decoder point-of-view, the absolute levels of the above and the left neighbor ($\boldsymbol{T}_A(c)$ and $\boldsymbol{T}_L(c)$) serve as the predictor $p = \max\{\boldsymbol{T}_A(c), \boldsymbol{T}_L(c)\}$, and one out of the three following cases applies. If the current absolute level $c$ is equal to one and the predictor is greater than 0 ($c = 1 \wedge p > 0$), the final absolute level is equal to the predictor value ($c = p$). The final absolute level is decreased by one when it is less or equal to the predictor value or it is unchanged otherwise, summarized as follows:

$$c = \begin{cases} p & c = 1 \wedge p > 0 \\ c - 1 & c \leq p \\ c & \text{otherwise.} \end{cases} \tag{5}$$

Note that the level prediction technique is equivalent to a modification of the binarization process, i.e., the level prediction swaps the bin string for $c = 1$ and $c = p$. Moreover, the level prediction is not applied when BDPCM is active for the current block.

### B. Block-Based DPCM

HEVC RExt specifies the residual DPCM (RDPCM) [17] coding technique that served as the trigger for developing the BDPCM coding tool in VVC. Specifically, RDPCM performs a sample-wise DPCM on the residual signal either in the horizontal or vertical direction, such that each residual row (or column for vertical direction) can be reconstructed at the decoder by just summing up the scaled DPCM residual levels (in case of lossy operation mode) over the given row (or column). Two RDPCM types exist in HEVC RExt, referred to as implicit and explicit RDPCM, and a conforming encoder can activate them separately via a high-level syntax flag in the sequence parameter set (SPS). As its name implies, explicit RDPCM requires additional signaling of the direction, and the application is confined to inter-predicted blocks only. On the other hand, implicit RDPCM does not require signaling of the direction, and its application is restricted to intra-predicted blocks with the corresponding prediction direction coupled to the intra prediction mode. Note that in contrast to the ordinary intra prediction, implicit RDPCM does not use any information outside the given block. For both RDPCM types, the encoder has to signal its usage for a transform block, i.e., component-wise, and only the 4:4:4 RExt profiles allow the usage of RDPCM. A primary motivation for introducing RDPCM in HEVC RExt is its use in lossless operation mode, while it turned out that RDPCM performs well for $R'G'B'$ screen content.

*1) Architecture:* In contrast to RDPCM in HEVC RExt, the block-based DPCM in VVC is considered an intra prediction mode, i.e., the signaling for its usage is within the prediction mode reconstruction process. The signaling requires for both luma and chroma two syntax elements each: the first syntax element flag $s_{BDPCM}$ indicates its usage, and the second syntax element flag $s_{BDPCM-DIR}$ indicates the horizontal or vertical direction whenever $s_{BDPCM} = 1$, where each syntax element employs a single dedicated context model for entropy coding. Applying the design in that way limits the DPCM process to the input samples in VVC from the encoder point-of-view, whereas the application of RDPCM is for all residual samples generated by the available prediction modes of HEVC RExt. From the decoder point-of-view, the syntax element $s_{tsm}$ can be inferred to be equal to 1 whenever the intra prediction mode is BDPCM ($s_{BDPCM} = 1$) as BDPCM is integrated into the TSM processing path and is only possible in conjunction with TSM. As in the implicit RDPCM design, the direction can be horizontal or vertical, depending on the BDPCM direction flag syntax element $s_{BDPCM-DIR} = 0$ or $s_{BDPCM-DIR} = 1$, respectively. The signaling for chroma blocks appears once in the bitstream and denotes the BDPCM usage for both chroma components, different from the RDPCM design in HEVC RExt. BDPCM is limited to the same block sizes as TSM, i.e., the coding block and the transform block sizes need to be the same, and there is no boundary filtering applied between adjacent BDPCM-predicted blocks within the deblocking filter (DBF) [28] processing. Finally, when BDPCM is active for the corresponding transform block, a single dedicated context model is used for coding the $s_{cbf}$ syntax element.

*2) Prediction and Reconstruction:* From an encoder perspective, the sample-wise prediction and reconstruction process of BDPCM is performed as follows. Let $s(x, y)$ and $\hat{s}(x, y)$ denote the original and reconstructed sample at the spatial location $(x, y)$, respectively. Furthermore, let $\tilde{r}(x, y)$ and $\hat{r}(x, y)$ denote the quantized and reconstructed, i.e., scaled DPCM residual samples at the coordinate $(x, y)$, respectively. Then, $\tilde{r}(x, y)$ and $\hat{s}(x, y)$ are given as follows, where $Q(\cdot)$ denotes the quantization operator:

$$\tilde{r}(x, y) = \begin{cases} Q(s(x, y)) & x = 0 \\ Q(s(x, y) - \hat{s}(x - 1, y)), & \text{otherwise} \end{cases} \tag{6}$$

$$\hat{s}(x, y) = \begin{cases} \hat{r}(x, y) & x = 0 \\ \hat{r}(x, y) + \hat{s}(x - 1, y), & \text{otherwise} \end{cases} \tag{7}$$

Note that for the encoder operation of BDPCM, the sample-wise prediction and quantization of the residual in eq. 6 is intertwined with the associated sample reconstruction process in eq. 7. From the decoder perspective, only the sample reconstruction process in eq. 7 is relevant. Due to the recursive nature of the DPCM loop, as specified in eq. 7, BDPCM introduces a serialization into the decoding process of one single row (column), which, however, can be mitigated by the fact that multiple rows or columns of a given block can be reconstructed in parallel. Note that the process described above specifies the horizontal direction mode of BDPCM; for the vertical direction of BDPCM, the process is the transposed version of that described above and therefore, operates on columns rather than rows. Although BDPCM acts as an intra prediction mode, no neighboring reference samples outside the given block are involved in the prediction process, and the original input sample values are unfiltered.

## IV. Extension of HEVC SCC Coding Tools

The JVET aims to achieve, at least, the same compression efficiency for screen content with the VVC profiles as the

HEVC Screen-Extended profiles by adopting the low-level coding tools of the HEVC SCC extensions. Although the naming and the coding tools' concepts are the same, stricter constraints to the architecture due to the inclusion in the Main profiles require refinement, avoiding excessive implementation cost. A special situation occurs for all three coding tools in VVC when the chroma separate tree (CST) is active, allowing different partitioning to the luma and the chroma coding blocks.

### *A. Intra Block Copy (IBC)*

Although the IBC concept existed before the HEVC standard, its introduction became reasonable with the appearance of screen content, where IBC provides significant compression efficiency improvements. As IBC should be an integral part of the Main profiles, a throughout optimization was necessary to reduce the implementation cost. The JVET achieved the target by restricting the reference area for memory requirements, so-called reference sample memory (RSM), enabling the possibility to realize IBC via an on-chip memory.

*1) Current Picture Referencing in HEVC:* The IBC coding tool appears in the HEVC SCC extensions as current picture referencing (CPR) and follows the coding path for inter prediction, although the IBC concept only requires the current frame. The main motivation behind the concept was the referencing structure, where the representation of the addressing mechanism to the reference samples can be as two-dimensional spatial vectors. Another benefit of such an architecture is that the integration of IBC requires the least changes to the specification and could ease the implementation burden, assuming manufacturers have already implemented the HEVC version 1. For the reason as mentioned above, CPR in the HEVC SCC extensions is a special inter prediction mode [29], resulting in the same syntax structure and almost the same decoding processes.

INTEGRATION INTO THE INTER PREDICTION PROCESS: As IBC (or CPR) is an inter prediction mode, an intra-only predicted slice has to become a predicted slice for allowing the usage of IBC [30]. When IBC is applicable, the coder must extend the reference picture lists by one entry for the pointer to the current picture, i.e., the current picture takes up to one picture-sized buffer of the shared decoded picture buffer (DPB). The IBC mode signaling is implicit, i.e., when the selected reference picture points to the current picture, the coding unit employs IBC [31]. Note that the reference samples within the IBC process are not filtered, which is in contrast to regular inter prediction, and the corresponding reference picture is a long-term reference. To minimize the memory requirement, the coder can immediately release the buffer after reconstructing the current picture [32]. Note that the coder may put a filtered version of the reconstructed picture back into the DPB as a short-term reference when it is a reference picture.

BLOCK VECTOR CODING: The referencing to a reconstructed area is via two-dimensional so-called block vectors (BV) as for inter prediction, and its prediction and coding reuse the motion vector (MV) prediction and coding of the inter prediction process. However, the luma BVs are in integer resolution rather than 1/4-th precision for the regular inter prediction mode. Consequently, the decoded motion vector differences (MVD) of BVs have to be left-shifted by two before adding to its predictor for the final BV reconstruction.

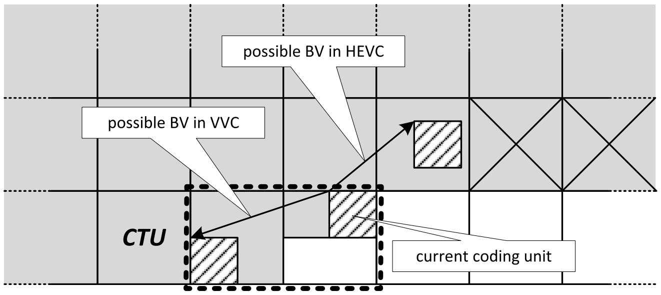


Fig. 4. The illustration summarizes the IBC concept in HEVC and VVC, where each square shape denotes a coding tree unit (CTU). The gray-shaded area denotes the already coded region, whereas the white-shaded area denotes the upcoming coding region. IBC in HEVC allows the gray-shaded region usage except for the two CTUs on the right above the current CTU for allowing Wavefront Parallel Processing (WPP). On the other hand, IBC in VVC only allows the CTU to the left of the current CTU as the reference area, denoted by the dotted frame.

DIFFERENCES TO INTER PREDICTION: Special handling was necessary for implementation and performance reasons, resulting in differences to the regular inter prediction mode, and they are as follows.

- The IBC reference samples are unfiltered, i.e., the reconstructed samples before the in-loop filtering processes, including DBF and Sample Adaptive Offset (SAO) [33], whereas the other inter prediction modes of HEVC employ filtered samples.
- There is no luma sample interpolation for IBC, and chroma sample interpolation is only necessary when the chroma BV is a non-integer when derived from the luma BV.
- A special case occurs when the chroma BV is in non-integer, and the reference block is near the boundary of an available region. Specifically, the surrounding reconstructed samples would be outside of the boundary to perform chroma interpolation. BVs pointing to a single next-to-border line [34] are not possible to avoid such cases.

*2) IBC Architecture in VVC:* The effective reference area for IBC in the HEVC SCC extensions is almost the whole already reconstructed area of the current picture, with some exceptions for parallel processing purposes. Fig. 4 illustrates the reference area for IBC in HEVC and the configuration in VVC, where only the coding tree unit (CTU) to the left of the current CTU served as the reference sample area at the beginning of the current CTU's reconstruction process. A drawback of the concept in HEVC is the requirement for additional memory in the DPB, for which hardware implementations usually employ external memory. Note that additional access to external memory comes with increased memory bandwidth, making the concept of using the DPB less attractive. VVC uses a fixed memory that can realize on-chip for IBC, significantly decreasing the complexity of implementing IBC in hardware architectures. Another significant modification address the

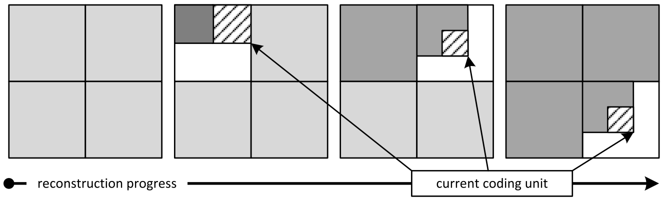


Fig. 5. The above figure illustrates the reference sample memory (RSM) update process at four intermediate times during the reconstruction process, where the light-gray shaded area denotes the reference samples of the left-neighboring CTU, the dark-gray shaded area denotes the reference samples of the current CTU, and the white-shaded area the upcoming coding region.

signaling concept departed from the integration within the inter prediction process as in the HEVC SCC extensions.

*3) Syntax and Semantics:* The IBC architecture in VVC forms a dedicated coding mode, where the IBC mode is the third prediction mode besides the intra and inter prediction modes. The bitstream carries the $s_{IBC}$ syntax element indicating the IBC mode for a coding unit when the block size is 64 × 64 or less. Consequently, the largest CU size that can utilize IBC is 64 × 64 to realize the continuous memory update mechanism of the reference sample memory (RSM). However, the reference sample addressing mechanism remains the same as in the HEVC SCC extensions by denoting a two-dimensional offset and reusing the inter prediction's vector coding processes. Another special case occurs when the CST is active, where the coder cannot derive chroma BVs from the luma BVs, resulting in the usage of IBC for the luma coding block only.

*4) Reference Area and Sample Memory:* The IBC design in VVC employs a fixed memory size of 128 × 128 for each color component for storing the reference samples [35], [36], enabling the possibility for an on-chip placement in hardware implementations. Note that the maximum CTU size in VVC is also 128 × 128, i.e., the reference sample memory (RSM) can hold samples of a single CTU when the maximum CTU size configuration is equal to 128 × 128. A special feature of the RSM is the continuous update mechanism replacing the reconstructed samples of the left-neighboring CTU with the reconstructed samples of the current CTU [37], [38]. Fig. 5 illustrates a simplified RSM example for the update mechanism at four intermediate times during the reconstruction process. The light-gray shaded area in Fig. 5 denotes the reference samples of the left-neighboring CTU, and the dark-gray shaded area the reference samples of the current CTU. At the first intermediate time, representing the beginning of the current CTU reconstruction, the RSM consists of reference samples of the left-neighboring CTU only. In the other three intermediate times, the reconstruction process has replaced samples of the left-neighboring CTU with the current CTU's variants. There is an implicit division of the RSM into four disjoint 64 × 64 areas where a reset of an area occurs when the coder processes the first coding unit that would lie in the corresponding area when mapping the RSM to the CTU, easing the hardware implementation efforts. Fig. 6 illustrates the continuous update concept of the RSM spatially, i.e., the left-neighboring CTU and the current CTU with the current coding unit. At the reconstruction time in the example, the processing has replaced the samples covered by the white-shaded area in the left-neighboring CTU with the gray-shaded area of the current CTU. Note that the RSM may contain more than a single left-neighboring CTU when the maximum CTU size is less than 128 × 128, resulting in the usage of multiple left-neighboring CTUs [39]. For example, when the maximum CTU size is equal to 32 × 32, the RSM may hold the samples of 15 left-neighboring CTUs.

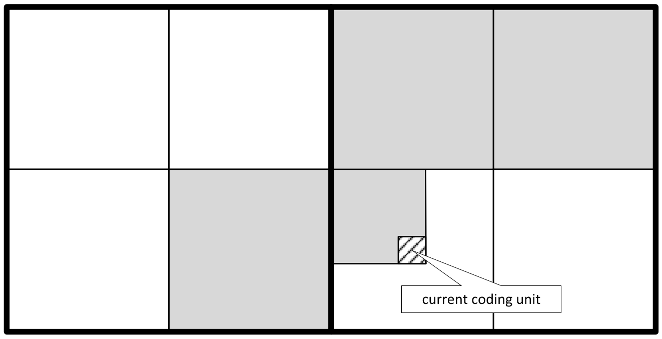


Fig. 6. The above figure illustrates the left-neighboring CTU and the current CTU denoting the effective reference area because of the RSM design and its continuous update mechanism. The gray shaded area covers the samples stored in the RSM, and the white shaded area covers replaced or unreconstructed samples.

*5) Block Vector Coding:* The BV coding employs the processes specified for inter prediction, but it uses more simplistic rules for candidate list construction. Specifically, the candidate list construction for inter prediction may consist of five spatial, one temporal, and six history-based candidates. Multiple candidate comparisons are necessary for history-based candidates, avoiding duplicate entries in the final candidate list. Additionally, the list construction may include pair-wise averaged candidates. In contrast to that, the IBC list construction process considers two spatial neighbors' BV and five history-based BVs (HBVP) [40] only, where only the first HBVP will compare with spatial candidates when added to the candidate list. While the regular inter prediction uses two different candidate lists, one for the merge mode and the other for the regular mode, the candidate list in IBC is for both cases [41]. However, the merge mode may use up to six candidates of the list, whereas the regular mode uses only the first two candidates. The block vector difference (BVD) coding employs the motion vector difference (MVD) process, resulting in a final BV of any magnitude. It also means that the reconstructed BV may point to an area outside of the reference sample area, requiring a correction by removing the absolute offset for each direction using the modulo operation with the RSM's width and height [42], [43].

### *B. Adaptive Color Transform (ACT)*

Typical end-consumer applications for video compression employ the $Y'C_BC_R$ color space representation because of compression efficiency and the color sensitivity of the human visual system, although the final color space for display is often $R'G'B'$. On the other hand, at the very high bit-rate end, the direct coding of content in the $R'G'B'$ domain provides higher fidelity than an external conversion from $R'G'B'$ to

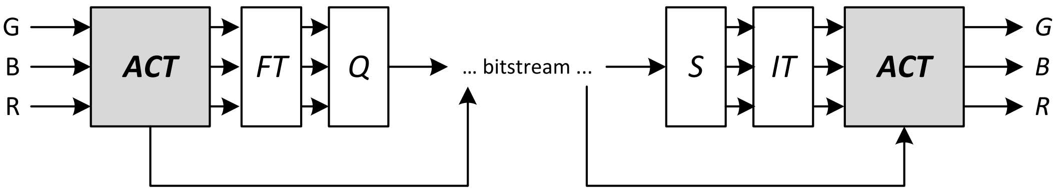


Fig. 7. The block diagram above denotes the ACT coding path when the input signal is in the $R'G'B'$ representation. ACT's application is before the transform coding stage, i.e., on the spatial residuals, and forward signaling of the application for each transform unit is necessary. Note that the coding of $R'G'B'$ content commonly employs the green component as luma because the green component usually contains most of the structural information.

$Y'C_BC_R$ prior to encoding and then back after the decoding due to rounding errors during the color transforms. When generating two rate-distortion curves, one for the compression using $R'G'B'$ and another using an external color space conversion to $Y'C_BC_R$ and back, there is a crossing point where the direct coding in $R'G'B'$ is more efficient, although at a high bit-rate operation point not typical for end-consumer applications [44]. In theory, an adaptive color space conversion embedded in the codec design enables the possibility to achieve a rate-distortion curve with at least the same compression efficiency as one out of the two rate-distortion curves mentioned above, depending on which one performs better at a given bit-rate. The HEVC RExt project introduced CCP [45] for the reason as mentioned above, as the development targeted professional applications. It turned out during the development of the SCC extensions for HEVC that ACT provides compression efficiency for 4:4:4 $R'G'B'$ screen content, even in the presence of CCP.

*1) ACT Architecture in HEVC:* ACT in the HEVC SCC extensions uses a transformation corresponding to the $Y'C_oC_g$ color space [46]–[48] as the alternative representation forward-adaptively for each transform unit that is coded in ACT mode, i.e., the application requires the luma transform block and the associated chroma transform blocks. Note that color space interpretation information in HEVC can appear as video usability information (VUI) metadata, but the VUI signaling does not affect the low-level decoding process. Thus, the ACT design implicitly assumes that the input color space is $R'G'B'$, although a configuration with $Y'C_BC_R$ and ACT active is possible (and it may provide coding gain for some video content).

HIGH-LEVEL SYNTAX: An HEVC Screen-Extended encoder can activate ACT in the PPS whenever the chroma format is 4:4:4. Since the ACT color transform is not orthonormal, quantization parameter (QP) offsets are applied to the transform blocks that are coded using ACT, to compensate for the norms of the synthesis basis functions of the color transform that are used in the ACT lossy coding case. The values of these offsets can be adjusted at the PPS and slice header levels of the syntax relative to default offset values of $(-5, -5, -3)$ for the three color components. Note that the above notation means that the QP offset is equal to $-5$ for the luma and the first chroma component, whereas it is $-3$ for the second chroma component, when not signaling offsets in the PPS and the slice header.

LOW-LEVEL SYNTAX: When ACT is activated, each transform unit includes an $s_{ACT}$ syntax element specifying whether ACT is active for that transform unit, but it only appears in the bitstream when at least one transform block of the transform unit has significant residuals, i.e., at least one out of three $s_{cbf}$ has to be equal to 1. Additionally, for intra-predicted coding units, the prediction unit has to be 2N × 2N and the chroma prediction mode has to be the same as the luma prediction mode for the $s_{ACT}$ syntax element to appear in the bitstream. Whenever the $s_{ACT}$ does not appear in the bitstream because of the conditions above, the decoder infers the value of $s_{ACT}$ to be equal to 0, i.e., the ACT is not in use for the corresponding transform unit. Another special case occurs for the lossless operation mode, where the ACT is not possible if the luma and the chroma bit depths are different. Finally, the entropy coding of $s_{ACT}$ employs a single dedicated context model.

COLOR TRANSFORMS: When ACT is applied to a transform unit, the encoder performs a conversion from the input to the alternative color space before the transform coding stage on the residuals, whereas the decoder performs the inverse color transform after the inverse spatial transform coding to reconstruct the residuals. Fig. 7 illustrates the corresponding block diagram for the ACT coding path and shows the location where ACT applies. The implementation in HEVC uses two different color transform variants: the $Y'C_oC_g$ representation for lossy coding [46], [48] and the $Y'C_oC_g$-$R$ representation [47], [48] for lossless coding. The following two equations summarize the $Y'C_oC_g$ concept for the lossy variant, where the eq. 8 denotes the forward and the eq. 9 the inverse transform.

$$\begin{bmatrix} Y' \\ C_o \\ C_g \end{bmatrix} = \frac{1}{4} \times \begin{bmatrix} -1 & 2 & 1 \\ 2 & 0 & -2 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} Y' \\ C_o \\ C_g \end{bmatrix} \tag{9}$$

For the lossless operation point, eq. 10 denotes the forward and eq. 11 the inverse transforms of the employed $Y'C_oC_g$-$R$ representation.

$$\begin{aligned} C_o &= R' - B' \\ t &= B' + (C_o \gg 1) \\ C_g &= G' - t \\ Y' &= t + (C_g \gg 1) \end{aligned} \tag{10}$$

$$\begin{aligned} t &= Y' - (C_g \gg 1) \\ G' &= C_g + t \\ B' &= t - (C_o \gg 1) \\ R' &= C_o + B' \end{aligned} \tag{11}$$

where $\gg$ indicates a right shift operation. Note that aside from possible differences in rounding, the $Y'$ component is the same for the two variants, while the $C_o$ and $C_g$ components are scaled up by a factor of 2 for $Y'C_oC_g$-$R$ in order to enable exact reversible transformation with minimal dynamic range expansion. Note that an external color space conversion before the encoding from $R'G'B'$ to $Y'C_oC_g$ is not lossless unless one increases the sample bit-depth by one for the chroma samples.

*2) Design Refinements for VVC:* The JVET adopted ACT into VVC, given its benefit for 4:4:4 $R'G'B'$ screen content and the fact that a vast quantity of screen content materials are in the 4:4:4 $R'G'B'$ representation [49], [50]. However, modifications were made to integrate ACT into the VVC architecture and to limit the implementation effort, especially for hardware implementations.

HIGH-LEVEL SYNTAX: Since the ACT coding tool applies only to the 4:4:4 chroma format, its activation flag in the SPS syntax is conditioned on the use of that chroma format. A further restriction applies that allows ACT only when the maximum transform size is 32 × 32 (i.e., not 64 × 64), in order to limit the maximum memory buffering requirements for decoder implementations. The limitation is reflected in the low-level syntax signaling. Unlike in HEVC, the QP offset values for lossy coding are not adjustable at the PPS and slice header levels, since such flexibility did not seem necessary.

LOW-LEVEL SYNTAX: The syntax element $s_{ACT}$ appears within the coding unit level in VVC instead of the transform unit level as in the HEVC SCC extensions. A direct consequence compared to the transform unit level is that a conditioning on $s_{cbf}$ does not exist anymore, leading to a possible state where the input signals to the inverse color transform can be insignificant. The $s_{ACT}$ appears in the syntax in two locations, one for the case when the coding unit employs intra prediction and another for when the coding unit employs inter prediction. For inter-prediction, the only constraint is the activation of the ACT coding tool, as specified in the SPS. However, in the case of intra prediction, $s_{ACT}$ only appears in the bitstream when the CST is not active and before the chroma intra prediction mode specification. This allows the inference of the chroma intra prediction mode, i.e., when $s_{ACT} = 1$, the chroma intra prediction mode is the same as the luma prediction mode, saving the signaling of the chroma intra prediction mode. Moreover, intra sub-partitioning (ISP) [51] and BDPCM signaling do not appear in the bitstream when $s_{ACT} = 1$ as their signaling is after the signaling of $s_{ACT}$. Consequently, the usage of ISP or BDPCM is not possible along the ACT coding path. Finally, as in HEVC, the syntax element $s_{ACT}$ uses a single dedicated context model for entropy coding. The restriction of $Y'C_oC_g$-$R$ to having equal bit depths for luma and chroma is implicit, as the use of unequal bit depths for luma and chroma was considered an unnecessary complication in VVC and was thus disallowed generally.

COLOR TRANSFORM: One of the main refinements of the ACT in VVC is the usage of the $Y'C_oC_g$-$R$ representation only, instead of having two variants as in the HEVC SCC extensions. Using a single color transform reduces implementation efforts and costs. The reason for using the $Y'C_oC_g$-$R$ representation is that it supports lossless as well as lossy coding, and its transformation equations use only simple shift and add-subtract operations. In VVC there is also no dedicated syntax element for signaling the lossless operation point as in HEVC. Instead, lossless operation is triggered by selection of the TSM coding path with a QP value equal to 4 or less. As in HEVC, VVC uses QP offsets for lossy coding in the ACT coding path, and the offsets are equivalent to the default values used in HEVC. Since there is a factor-of-two scaling difference for chroma in $Y'C_oC_g$-$R$ relative to ordinary $Y'C_oC_g$ (as noted above) and since a difference of 6 in the value of QP corresponds to a factor of two in quantization step size, the value of the QP offsets for the chroma differ by 6 from those in HEVC and are thus equal to $(-5, 1, 3)$.

### C. Palette Mode Coding

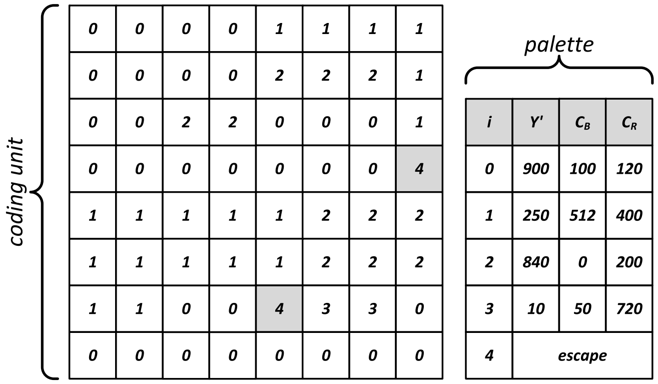
For distinct colored content, typical for computer-generated content with large amounts of text and simple graphics, the palette mode provides improved compression efficiency over regular block-based prediction and transform coding of the residuals. The palette mode coding path starts in HEVC and VVC for a coding unit and employs the hybrid video coding architecture's entropy coding engine, preserving the block-based concept and using it to its advantage when the distinct colored area is only partial.

*1) Palette Mode Architecture in HEVC:* The palette mode in the HEVC SCC extensions [10] supports coding units sized 32 × 32 and smaller and comprises two parts: the coding of the table denoting the distinct samples, i.e., the palette, and the coding of the index for each spatial position covered by the coding unit. Fig. 8 illustrates an example for an 8 × 8 coding unit and the corresponding palette consisting of four sample entries and a single escape entry. A palette entry consists of the index within the palette and the corresponding sample, where a sample is a triplet for non-monochrome video formats.

PALETTE SIGNALING: Each coding unit employing the palette mode uses its own palette, and the signaling comprises a prediction due to the high correlation among the palettes in neighboring regions. The predictor stores the palette information of already used palettes, and the construction of the palette for the current coding unit uses either a flag indicating the reuse of a predictor's entry or adds a new sample to the palette when the sample is not in the predictor's list. For the latter case, the predictor will then be updated with the palette, followed by the predictor's unused elements until the predictor reaches its maximum size. The maximum size for the palette is 64 and is 128 the predictor in the HEVC SCC extensions, where smaller values are possible via the signaling within the SPS.

ESCAPE ENTRY: The escape entry refers to a sample that is not within the palette, typically for cases where the sample appears infrequently. Whenever the index to the escape entry appears, the encoder transmits the sample directly in the bitstream, followed by adding the new sample to the palette while still keeping the escape entry as the last entry.

SAMPLE CODING: After finishing the palette construction, the coding of the spatial locations covered by the coding unit starts using either the horizontal or vertical scanning pattern, as illustrated in Fig. 9. For each scanning position,



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 2 | 2 | 2 | 1 |
| 0 | 0 | 2 | 2 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |
| 1 | 1 | 0 | 0 | 4 | 3 | 3 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| *i* | *Y'* | $C_B$ | $C_R$ |
|---|---|---|---|
| 0 | 900 | 100 | 120 |
| 1 | 250 | 512 | 400 |
| 2 | 840 | 0 | 200 |
| 3 | 10 | 50 | 720 |
| 4 | escape | | |

Fig. 8. The above figure shows an example for the palette mode coding in HEVC/VVC for an 8 × 8 coding unit (on the left) and the associated palette (on the right) consisting of five entries. Each spatial position of the coding unit holds the palette's index, with the fifth entry of the palette denoting the escape value.

the encoder signals the palette index using run-length coding with a classification in two types: *copy index* and *copy above*, where *copy index* represents the direct signaling of the index to the palette entry in the bitstream. On the other hand, the indices are the same as those of the above row (or left column) for the horizontal (or vertical) scanning pattern for the *copy above* mode. The samples' coding takes place in multiple scanning passes, as for the residual coding stages in VVC, where the first coding pass signals the indices for scanning positions using the copy index type. The second coding pass consists of context-coded bins denoting the type and run length information. Note that the number of indices is within the bitstream before the first coding pass allowing a decoder to know how many indices are within the bitstream for the current coding unit. Finally, the third scanning pass specifies the samples for scanning positions with an index equals to the escape entry. While the first and third scanning passes use bypass mode only for the corresponding syntax elements, the second coding pass employs context models for entropy coding.

*2) Design Refinements for VVC:* The palette mode architecture in VVC forms a dedicated coding mode next to the IBC, intra, and inter prediction modes. Like IBC and ACT, the palette coding mode experienced refinements either to reduce the implementation cost or to align the coding tool to the VVC architecture, or both.

CHROMA SEPARATE TREE: The palette coding mode for CST is independently for each color channel (the luma channel comprises of the luma component, and the chroma channel comprises of the two chroma components) [52]. Specifically, the palette's entries for luma consist of a single value denoting the sample, whereas the palette's entries for chroma consist of two values denoting the sample, and each channel employs its own palette predictor.

SAMPLE CODING: The palette sample coding in VVC [53] employs a sub-shape concept similar to that of the residual coding stage, where a sub-shape covers 16 consecutive scanning positions. Fig. 9 illustrates the concept in cooperation with the scanning pattern, where the gray-shaded area and the white-shaded area each forms a sub-shape within the coding unit. Each sub-shape forms a self-contained coding stage, i.e., the reconstruction process can starts when the shape's parsing process finishes. That is different from the case in the HEVC SCC extensions, where the reconstruction can start only when the parsing process for the coding unit finishes due to the grouping of context-coded and bypass-coded bins, as for the transform coefficient level coding in HEVC/VVC. The modification improves the entropy coding's throughput, and syntax elements data can be released after finishing a sub-shape rather than after the whole coding unit. The palette sample coding in VVC performs in multiple scanning passes for each sub-shape, where the first coding stage transmits the context-coded $s_{copy}$ syntax element, indicating whether the current scanning position employs the same type and index as the preceding scanning position. Whenever it is not the case, i.e., $s_{copy} = 0$, a further syntax element follows, indicating the type for the current scanning position. For run-length coding, the processing of values for the *copy index* type is in a dedicated coding pass using the truncated binary representation, i.e., using the entropy coding engine's bypass mode. Likewise, i.e., in the entropy coding engine's bypass mode, the processing of the quantized escape samples forms a further coding pass using $5^{th}$th-order Exp-Golomb code.

SMALLEST CHROMA INTRA PREDICTION UNIT (SCIPU): The SCIPU [54] concept avoids small chroma coding units in the 4:2:0 and the 4:2:2 chroma formats by using different partitioning for luma and chroma, similar to the CST concept. In SCIPU conditions, the palette mode is only active for the luma component, and the signaling is the same as for the luma channel in the CST mode with a joint update of the predictor, i.e., for both luma and chroma. As the palette mode usage is for luma only, the processing inserts the default value $2^{\text{bit-depth}-1}$ as new values for chroma before updating the palette predictor in the SCIPU case [55].

PREDICTOR INITIALIZATION: A reset of the predictor occurs at the beginning of each slice and tile in VVC, whereas the encoder can signal a set of pre-defined predictor entries in the PPS or the SPS for initialization in the HEVC SCC extensions.

PALETTE AND PREDICTOR SIZE: In contrast to the configurable maximum palette and predictor size in the HEVC SCC extensions, one cannot adjust the maximum sizes in VVC. For the palette size, the maximum is equal to 31 when operating in the single tree mode, whereas the maximum is equal to 15 in the CST mode. The maximum predictor size is equal to 63 in the single tree mode, and the value is equal to 31 in the CST mode. Note that the maximum sizes in CST mode are almost half the maximum in the single tree case due to two predictors and independent palettes [56].

MAXIMUM CODING UNIT SIZE: The palette mode supports coding units sized 64 × 64 or less excluding coding units covering less than or equal to 16 spatial positions. The latter may appear when using ISP, and the exclusion of the small coding units reduces the complexity and latency in the pipeline.

## V. EXPERIMENTAL RESULTS

The compression efficiency of VVC for screen content is superior to that of the HEVC Main 10 and the Screen-Extended Main 10 profiles, as the following results prove. This section starts with the experimental setup before

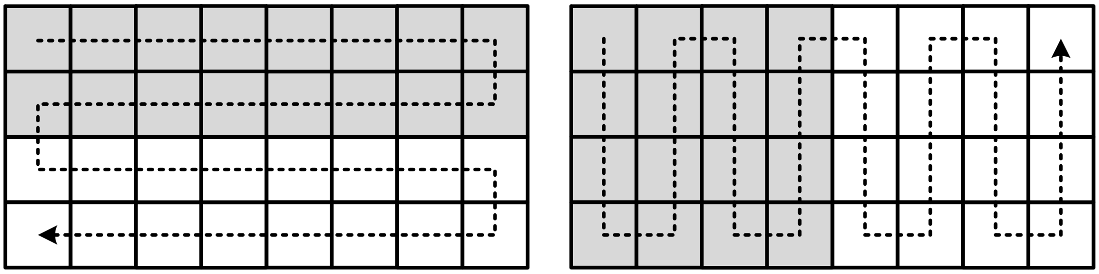


Fig. 9. The figure above illustrates the two scanning patterns for the palette mode, exemplarily for an 8 × 4 coding unit, and the partitioning of the coding unit into sub-shapes for entropy coding purposes in the VVC implementation.

discussing and analyzing the evaluation outcome for the 4:2:0 and 4:4:4 test sets separately. Both subsections consist of additional coding tools evaluation for each of the discussed screen content low-level coding tools in this paper.

### A. Experimental Setup

*1) Test Environment:* For the conducted experiments, the configurations followed the two different VVC Common Test Conditions (CTC): the JVET CTC [57] and the non-4:2:0 CTC [58]. All test sequences belonging to the JVET CTC test set are in the 4:2:0 chroma sampling format, whereas the non-4:2:0 CTC specifies two 4:4:4 test sets and one 4:2:2 test set, where the latter does not include screen content sequences as there are no relevant applications having screen content in the 4:2:2 chroma format. Only the sequence groups labelled as Class F and TGM (Text and Graphics with Motion) of the 4:2:0 test set are from interests since they represent screen content signals. On the other hand, one of the 4:4:4 test set consists of camera-captured test sequences only and is also not relevant for the conducted experiments. Finally, a further classification into $Y'C_BC_R$ and $R'G'B'$ content exists for the 4:4:4 screen content test set.

*2) Software and Configurations:* The experiments employed the HEVC reference software implementation HM-16.21 as a representative for an HEVC compliant encoder using the Main 10 profile and the HM-16.21 + SCC-8.8 software package as a corresponding representative for an HEVC compliant encoder using the Screen-Extended Main 10 profile. For VVC, the experiments used the VVC Test Model (VTM) version 9.0 reference software implementation, and the configurations of the coding tools for all tested encoders are according to the specification in the corresponding CTCs. Experiments and results presented in the following subsections are only for the Random-Access (RA) configuration as RA represents the most widely used application. Specifically, the RA configuration allows for random access, enabling essential features such as channel switching for broadcasting, limited error resiliency, and seeking. The encoder has to transmit intra random access point (IRAP) pictures to realize random access points within the bitstream, and the CTCs specify the appearance of IRAP pictures at a constant interval of about one second.

*3) Evaluation:* Both employed CTCs specify the usage of a four-point Bjøntegaard Delta (BD) bit-rate calculation (BD-rate) [59], which consequently requires four operation points. The presented BD-rate percentage values are bit-rate savings or overhead using the Peak-Signal-to-Noise-Ratio (PSNR) metric. For simplicity of interpretation, a positive number indicates a bit-rate saving. Each encoder software implementation encoded a single test sequence four times using different QPs to generate the four operation points. Specifically, the used QP values were 22, 27, 32, and 37 for all employed encoders according to both CTCs.

### B. Results for the 4:2:0 Test Set

Table I summarizes the luma BD-rate values (Luma BDR), encoding time (EncT), and decoding time (DecT) for VTM-9.0, representing the VVC Main 10 profile, relative to HM-16.21, representing the HEVC Main 10 profile and HM-16.21 + SCC-8.8, representing the HEVC Screen-Extended Main 10 profile. Specifically, VTM-9.0 operation points always served as the test candidate for the BD-rate calculation, whereas either HM-16.21 or HM-16.21 + SCC-8.8 operation points served as the anchor. As expected, the bit-rate savings relative to the HEVC version 1 (Main 10 profile) are significant, with an average bit-rate savings of about 41% for Class F and 60% for TGM. Interestingly, there are still significant bit-rate savings relative to the HEVC SCC extensions (Screen-Extended Main 10 profile), with an average bit-rate savings of about 25% for Class F and 19% for TGM. Although the bit-rate savings seem to be consistent for all test sequences, different aspects may contribute to the outcome. For example, the test sequence *BasketballDrillText* of Class F consists of a small fraction of screen content only, leading to the assumption that the main part of the bit-rate savings is probably due to coding tools suitable for camera-captured content in VVC. On the other hand, the *ArenaOfValor* animation sequence also has a relatively high bit-rate savings, i.e., VTM shows a significant bit-rate savings relative to HM and SCC. The observation is interesting as screen content coding tools usually achieve lower compression efficiency for animation content than for other screen content types [12].

Table II summarizes the BD-rate values for a coding tool evaluation in VTM-9.0, i.e., an IBC and TSM+ disabled configuration served as the anchor while a configuration, having either IBC or TSM+ enabled, served as the test candidate. Note that TSM+ stands for a configuration with TSM, TSRC, and BDPCM enabled, where the encoder can apply TSM for transform block sizes up to 32 × 32. The outcome shows that both TSM+ and IBC improve the compression efficiency at a similar performance level and that the two coding tools are not mutually exclusive. Specifically, the configuration

TABLE I

BD-RATE VALUES, ENCODING, AND DECODING TIMES OF VTM 9.0 RELATIVE TO HM 16.21 AND HM-16.21 + SCC-8.8 FOR 4:2:0 SCREEN CONTENT

| Sequence | Luma — $Y'$ | Chroma — $C_B$ | Chroma — $C_R$ | EncT | DecT |
|---|---|---|---|---|---|
| **VTM-9.0 relative to HM-16.21** | | | | | |
| BasketballDrillText | 33.56% | 39.90% | 41.32% | 1283% | 166% |
| ArenaOfValor | 33.25% | 34.63% | 31.36% | 863% | 160% |
| SlideEditing | 54.61% | 56.49% | 59.17% | 337% | 120% |
| SlideShow | 44.38% | 47.65% | 51.25% | 422% | 137% |
| **Averaged F** | **41.45%** | **44.67%** | **45.77%** | **629%** | **145%** |
| FlyingGraphics | 51.64% | 56.28% | 59.48% | 1236% | 173% |
| Desktop | 70.32% | 69.43% | 69.27% | 346% | 122% |
| Console | 68.71% | 68.27% | 66.00% | 567% | 139% |
| ChineseEditing | 52.23% | 54.71% | 54.43% | 377% | 137% |
| **Averaged TGM** | **60.72%** | **62.17%** | **62.30%** | **918%** | **142%** |
| **VTM-9.0 relative to HM-16.21+SCC-8.8** | | | | | |
| BasketballDrillText | 31.78% | 35.95% | 37.58% | 696% | 151% |
| ArenaOfValor | 31.62% | 32.06% | 28.10% | 589% | 136% |
| SlideEditing | 7.12% | 11.86% | 16.77% | 629% | 97% |
| SlideShow | 30.93% | 34.25% | 38.01% | 495% | 113% |
| **Averaged F** | **25.36%** | **28.53%** | **30.11%** | **598%** | **123%** |
| FlyingGraphics | 28.22% | 29.81% | 32.50% | 657% | 170% |
| Desktop | 7.62% | 11.55% | 10.22% | 743% | 102% |
| Console | 27.07% | 26.80% | 25.26% | 633% | 124% |
| ChineseEditing | 14.73% | 20.53% | 21.90% | 912% | 114% |
| **Averaged TGM** | **19.41%** | **22.17%** | **22.47%** | **531%** | **125%** |

TABLE II

BD-RATE VALUES, ENCODING, AND DECODING TIMES OF TSM+ AND IBC IN VTM 9.0 RELATIVE TO A CONFIGURATION WITH BOTH CODING TOOLS DISABLED FOR 4:2:0 SCREEN CONTENT

| Tool | Class | Luma — $Y'$ | Chroma — $C_B$ | Chroma — $C_R$ | EncT | DecT |
|---|---|---|---|---|---|---|
| TSM+ | F | 7.64% | 7.11% | 7.36% | 106% | 100% |
| IBC | F | 9.66% | 10.20% | 10.36% | 115% | 100% |
| TSM+ | TGM | 24.64% | 21.87% | 22.17% | 110% | 93% |
| IBC | TGM | 31.42% | 30.43% | 30.82% | 97% | 95% |
| **All Tools** | F | **15.40%** | **14.83%** | **15.14%** | **121%** | **98%** |
| **All Tools** | TGM | **41.75%** | **38.70%** | **39.21%** | **109%** | **88%** |

with IBC and TSM+ enabled shows a significantly higher compression efficiency improvement than the configurations where only a single coding tool is enabled. The outcome of both comparisons shows significantly higher encoding and decoding times with a high variation for the encoding time depending on the test sequence. While the increased decoding time is supposedly due to the more advanced architecture of VVC, the increased encoding time is reducible when it comes to practical encoder implementations, e.g., via a screen content detection algorithm and executing the rate-distortion optimization (RDO) only for the screen content coding tools.

## *C. Results for the 4:4:4 Test Set*

Table III summarizes the BD-rate values for the 4:4:4 test set using the way of comparison as for the results in Table I, i.e., VTM-9.0 is the test candidate, and either HM-16.21 or HM-16.21 + SCC-8.8 is the anchor. Note that the used HM-16.21 configuration represents the Main 4:4:4 profile of HEVC version 2 as there was no support for the 4:4:4 chroma format in version 1. The observation is similar to that of the 4:2:0 test set, i.e., VTM-9.0 provides on average about 13% bit-rate savings relative to HM-16.21 + SCC-8.8 and 53% relative to HM-16.21. Notably, one observes a relatively significant bit-rate savings for the *Animation* class of about 33% for $Y'C_BC_R$ content. However, compared to a configuration with all low-level screen content coding tools disabled, one observes a bit-rate savings of about 30%, indicating that the improved compression efficiency results from the coding tools designed for camera-captured content. The conclusion from that observation is that although the classification for

TABLE III

BD-RATE VALUES, ENCODING, AND DECODING TIMES OF VTM 9.0 RELATIVE TO HM 16.21 AND HM-16.21 + SCC-8.8 FOR 4:4:4 SCREEN CONTENT

| Class | Colour Space | Luma<br>$Y'$ or $G'$ | Chroma<br>$C_B$ or $B'$ | Chroma<br>$C_R$ or $R'$ | EncT | DecT |
|---|---|---|---|---|---|---|
| **VTM-9.0 relative to HM-16.21** | | | | | | |
| TGM 1080p | $Y'C_BC_R$ | 64.36% | 66.14% | 66.03% | 510% | 125% |
| TGM 720p | $Y'C_BC_R$ | 52.24% | 54.88% | 58.21% | 368% | 128% |
| Animation | $Y'C_BC_R$ | 33.43% | 39.72% | 41.06% | 753% | 144% |
| Mixed Content | $Y'C_BC_R$ | 49.07% | 53.37% | 53.52% | 371% | 129% |
| TGM 1080p | $R'G'B'$ | 65.67% | 64.92% | 64.89% | 595% | 130% |
| TGM 720p | $R'G'B'$ | 56.48% | 51.33% | 53.70% | 478% | 133% |
| Animation | $R'G'B'$ | 43.13% | 37.38% | 37.60% | 1032% | 155% |
| Mixed Content | $R'G'B'$ | 53.43% | 48.78% | 48.34% | 494% | 137% |
| **Averaged** | | **53.29%** | **53.10%** | **54.03%** | **534%** | **134%** |
| **VTM-9.0 relative to HM-16.21+SCC-8.8** | | | | | | |
| TGM 1080p | $Y'C_BC_R$ | 13.29% | 9.80% | 9.82% | 746% | 113% |
| TGM 720p | $Y'C_BC_R$ | 14.17% | 10.64% | 10.43% | 720% | 108% |
| Animation | $Y'C_BC_R$ | 26.63% | 25.84% | 27.73% | 589% | 125% |
| Mixed Content | $Y'C_BC_R$ | 19.14% | 17.71% | 18.66% | 783% | 103% |
| TGM 1080p | $R'G'B'$ | 6.47% | 8.55% | 9.36% | 777% | 118% |
| TGM 720p | $R'G'B'$ | 5.67% | 8.97% | 7.45% | 812% | 115% |
| Animation | $R'G'B'$ | 14.79% | 22.39% | 20.21% | 734% | 139% |
| Mixed Content | $R'G'B'$ | 8.37% | 11.55% | 11.56% | 853% | 109% |
| **Averaged** | | **13.04%** | **13.72%** | **13.67%** | **750%** | **115%** |

TABLE IV

BD-RATE VALUES, ENCODING, AND DECODING TIMES OF TSM+, IBC, PALETTE, AND ACT IN VTM 9.0 RELATIVE TO A CONFIGURATION WITH THE TOOLS DISABLED FOR 4:4:4 SCREEN CONTENT

| Tool | Colour Space | Luma<br>$Y'$ or $G'$ | Chroma<br>$C_B$ or $B'$ | Chroma<br>$C_R$ or $R'$ | EncT | DecT |
|---|---|---|---|---|---|---|
| TSM+ | $Y'C_BC_R$ | 13.71% | 14.60% | 15.28% | 107% | 95% |
| IBC | $Y'C_BC_R$ | 16.15% | 16.56% | 16.67% | 104% | 97% |
| Palette | $Y'C_BC_R$ | 17.64% | 20.39% | 21.46% | 99% | 93% |
| TSM+ | $R'G'B'$ | 15.50% | 16.49% | 16.47% | 108% | 94% |
| IBC | $R'G'B'$ | 14.15% | 14.88% | 14.90% | 107% | 96% |
| Palette | $R'G'B'$ | 21.37% | 20.70% | 20.74% | 99% | 90% |
| ACT | $R'G'B'$ | 11.07% | 3.02% | 4.36% | 103% | 103% |
| **All Tools** | $Y'C_BC_R$ | **33.22%** | **35.77%** | **36.86%** | **118%** | **90%** |
| **All Tools** | $R'G'B'$ | **38.62%** | **37.16%** | **38.22%** | **124%** | **91%** |

animation content is as screen content, its signal characteristic is presumably closer to camera-captured content statistics.

Finally, Table IV summarizes the BD-rate values for a tool evaluation, i.e., again, a configuration without the screen content coding tools enabled served as the anchor. On top of disabling TSM+ and IBC, one has to disable the palette mode and ACT for the $R'G'B'$ test sequences. Note that when enabling ACT, one has to disable the CST and not set the maximum allowed transform size to 64 × 64. Consequently, the anchor and all other experiments without ACT enabled had the CST enabled without the maximum allowed transform size restriction. The compression efficiency improvement for TSM+, IBC, and the palette mode are in a similar range, with a slightly higher averaged bit-rate savings for the palette mode, followed by IBC and then TSM+. A notable outcome is the decoding times that are lower than the configuration without screen content coding tools enabled. The reason for that is a shift in operation points due to more efficient representation in RDO-sense, i.e., the bitstreams generated by the used fixed QPs have lower bit-rates, resulting in a faster parsing in reconstruction at the decoder side.

## VI. CONCLUSION

The specification of low-level coding tools for screen content in the first VVC version enables the prospect to represent screen content signals more efficiently than its predecessor

HEVC. That statement remains valid even compared to the HEVC SCC extensions that have dedicated support for screen content signals. The outcome is thanks to the inclusion and refinement of the HEVC SCC extensions low-level coding tools IBC, ACT, and the palette mode into the VVC design and the extension of the TSM coding path by TSRC and BDPCM. Although the overall compression efficiency improvement when enabling all low-level coding tools is slightly less than the sum of each tool's improvement, the outcome proves that they are not mutually exclusive. Consequently, it also proves that the package of screen content coding tools in VVC is well-designed. There is a clear argument to expect that VVC becomes the main choice for applications having fully or partially screen content, given the compression efficiency improvements over HEVC.

## Acknowledgment

The authors would like to thank all the experts of the involved standardization organizations, which cannot be individually mentioned here. Versatile Video Coding is the result of their joint efforts and contributions.

## References

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

[2] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC)," *Proc. IEEE*, early access, Jan. 19, 2021, doi: 10.1109/JPROC.2020.3043399.

[3] B. Bross *et al.*, "Overview of the versatile video coding (VVC) standard and its applications," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[4] *High Efficiency Video Coding*, Standard Rec. ITU-T H.265 and ISO/IEC 23008- 2,, Oct. 2014.

[5] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[6] T. Nguyen, B. Bross, H. Schwarz, D. Marpe, and T. Wiegand, "Residual coding for transform skip mode in versatile video coding," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2020, pp. 83–92.

[7] M. Abdoli, F. Henry, P. Brault, F. Dufaux, P. Duhamel, and P. Philippe, "Intra block-DPCM with layer separation of screen content in VVC," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 3162–3166.

[8] X. Xu *et al.*, "Intra block copy in HEVC screen content coding extensions," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 409–419, Dec. 2016.

[9] L. Zhang *et al.*, "Adaptive color-space transform in HEVC screen content coding," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 446–459, Dec. 2016.

[10] W. Pu *et al.*, "Palette mode coding in HEVC screen content coding extension," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 420–432, Dec. 2016.

[11] M. Mrak and J. Xu, "Improving screen content coding in HEVC by transform skipping," in *Proc. 20th Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2012, pp. 1209–1213.

[12] J. Xu, R. Joshi, and R. A. Cohen, "Overview of the emerging HEVC screen content coding extension," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 50–62, Jan. 2016.

[13] S. Liu, X. Xu, S. Lei, and K. Jou, "Overview of HEVC extensions on screen content coding," *APSIPA Trans. Signal Inf. Process.*, vol. 4, pp. 1–12, Sep. 2015.

[14] 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.

[15] R. Joshi, S. Liu, G. Sullivan, J. Xu, and Y. Ye, *HEVC Screen Content Coding Draft Text 4*, document JCTVC-U1005 of the 21st JCT-VC Meeting, 2015.

[16] B. Li, J. Xu, G. J. Sullivan, Y. Zhou, and B. Lin, *Adaptive Motion Vector Resolution for Screen Content*, document JCTVC-S0085 of the 19th JCT-VC Meeting, 2014.

[17] D. Flynn *et al.*, "Overview of the range extensions for the HEVC standard: Tools, profiles, and performance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 4–19, Jan. 2016.

[18] M. Naccari, S. G. Blasi, M. Mrak, and E. Izquierdo, "Improving inter prediction in HEVC with residual DPCM for lossless screen content coding," in *Proc. Picture Coding Symp. (PCS)*, Dec. 2013, pp. 361–364.

[19] A. Khairat, T. Nguyen, M. Siekmann, D. Marpe, and T. Wiegand, "Adaptive cross-component prediction for 4: 4: 4 high efficiency video coding," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 3734–3738.

[20] W.-J. Chien *et al.*, "Motion vector coding and block merging in versatile video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[21] C. Rosewarne, K. Sharman, and D. Flynn, *Common Test Conditions and Software Reference Configurations for HEVC Range Extensions*, document JCTVC-P1006 of the January 2014 JCT-VC Meeting, 2014.

[22] H. Yu, R. Cohen, K. Rapaka, and J. Xu, *Common Test Conditions for Screen Content Coding*, document JCTVC-Z1015 of the January 2017 JCT-VC Meeting, 2017.

[23] T. Nguyen, B. Bross, P. Keydel, H. Schwarz, D. Marpe, and T. Wiegand, "Extended transform skip mode and fast multiple transform set selection in VVC," in *Proc. Picture Coding Symp. (PCS)*, Nov. 2019, pp. 1–5.

[24] H. Schwarz *et al.*, "Quantization and entropy coding in the versatile video coding (VVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Apr. 9, 2021, doi: 10.1109/TCSVT.2021.3072202.

[25] T. Nguyen, H. Schwarz, H. Kirchhoffer, D. Marpe, and T. Wiegand, "Improved context modeling for coding quantized transform coefficients in video compression," in *Proc. 28th Picture Coding Symp.*, Dec. 2010, pp. 378–381.

[26] T. Nguyen *et al.*, "Transform coding techniques in HEVC," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 978–989, Dec. 2013.

[27] H. Schwarz, T. Nguyen, D. Marpe, and T. Wiegand, "Hybrid video coding with trellis-coded quantization," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2019, pp. 182–191.

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

[29] X. Xu, S. Liu, and S. Lei, *On Unification of Intra Block Copy and Inter-Picture Motion Compensation*, document JCTVC-Q0132 of the 17th JCT-VC Meeting, 2014.

[30] C. Pang *et al.*, *Non-CE2: Intra Block Copy and Inter Signalling Unification*, document JCTVC-T0227 of the 20th JCT-VC Meeting, 2015.

[31] X. Xu, S. Liu, and S. Lei, *On Reference Picture List Construction for Intra Block Copy*, document JCTVC-U0113 of rhe 21st JCT-VC Meeting, 2015.

[32] X. Xu *et al.*, *On Storage of Filtered and Unfiltered Current Decoded Pictures*, document JCTVC-U0181 of the 21st JCT-VC Meeting, 2015.

[33] C.-M. Fu *et al.*, "Sample adaptive offset in the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1755–1764, Dec. 2012.

[34] X. Xu, S. Liu, and S. Lei, *On Chroma Motion Vector Derivation for Intra Block Copy*, document JCTVC-U0077 of the 21st JCT-VC Meeting, 2015.

[35] X. Xu, X. Li, and S. Liu, *CE8: CPR Mode with Local Search Ranges (Test CE8.3.1 and CE8.3.2)*, document JVET-L0293 of the 12th JVET Meeting, 2018.

[36] X. Xu, X. Li, and S. Liu, "Current picture referencing in versatile video coding," in *Proc. IEEE Conf. Multimedia Inf. Process. Retr. (MIPR)*, Mar. 2019, pp. 26–31.

[37] X. Xu, X. Li, S. Liu, and E. Chai, *CE8: CPR Reference Memory Reuse without Increasing Memory Requirement (CE8.1.2a and CE8.1.2d)*, document JVET-M00407 of the 13th JVET Meeting, 2019.

[38] X. Xu, X. Li, and S. Liu, "Intra block copy in versatile video coding with reference sample memory reuse," in *Proc. Picture Coding Symp. (PCS)*, Nov. 2019, pp. 1–5.

[39] X. Xu, X. Li, and S. Liu, *Non-CE8: IBC Search Range Increase for Small CTU Sizes*, document JVET-N0384 of the 14th JVET Meeting, 2019.

[40] X. Xu, S. Liu, T.-D. Chuang, and S. Lei, "Block vector prediction for intra block copying in HEVC screen content coding," in *Proc. Data Compress. Conf.*, Apr. 2015, pp. 273–282.

[41] X. Xu *et al.*, *CE8-Related: Combination Test of JVET-N0176/JVET-N0317/JVET-N0382 on Simplification of IBC Vector Prediction*, document JVET-N0843 of the 14th JVET Meeting, 2019.

[42] J. Xu *et al.*, *Bitstream Conformance with a Virtual IBC Buffer Concept*, document JVET-O1170 of the 15th JVET Meeting, 2019.

[43] J. Xu *et al. Non-CE8: Alternative IBC Virtual Buffer Setting to Avoid Reference Sample Wrapping Around*, document JVET-P1018 of the 16th JVET Meeting, 2019.

[44] D. Marpe, H. Kirchhoffer, V. George, P. Kauff, and T. Wiegand, "Macroblock-adaptive residual color space transforms for 4: 4: 4 video coding," in *Proc. Int. Conf. Image Process.*, Atlanta, GA, USA, Oct. 2006, pp. 3157–3160.

[45] 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.

[46] H. Malvar and G. Sullivan, *Transform, Scaling & Color Space Impact of Professional Extensions*, document JVT-H031 of the May 2003 JVT Meeting, 2003.

[47] H. Malvar and G. Sullivan, *YCoCg-R: A Color Space with RGB Reversibility and Low Dynamic Range*, document JVT-I014 of the July 2003 JVT PExt AHG Meeting, 2003.

[48] H. S. Malvar, G. J. Sullivan, and S. Srinivasan, "Lifting-based reversible color transformations for image compression," *Proc. SPIE*, vol. 7073, pp. 44–53, Sep. 2008, Art. no. 707307, doi: 10.1117/12.797091.

[49] X. Xiu, Y.-W. Chen, T.-C. Ma, H.-J. Jhu, and X. Wang, *Support of Adaptive Color Transform for 444 Video Coding in VVC*, document JVET-P0517 of the 16th JVET Meeting, 2019.

[50] X. Xiu *et al.*, *ACT Color Conversion for Both Lossless and Lossy Coding*, document JVET-Q0510 of the 17th JVET Meeting, 2020.

[51] S. De-Luxan-Hernandez *et al.*, "An intra subpartition coding mode for VVC," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 1203–1207.

[52] Y.-H. Chao *et al.*, *CE8-2.1: Palette Mode in HEVC*, document JVET-O0119 of the 15th JVET Meeting, 2019.

[53] Y.-H. Chao, C.-H. Hung, W.-J. Chien, T. Hsieh, and M. Karczewicz, *CE8-1.3: Line-based CG Palette Mode*, document JVET-P0077 of the 16th JVET Meeting, 2019.

[54] Z.-Y. Lin *et al.*, *CE3-2.1.1 and CE3-2.1.2: Removing 2×2, 2×4, and 4×2 Chroma CBs*, document JVET-O0050 of the 15th JVET Meeting, 2019.

[55] R.-L. Liao *et al.*, *CE2-Related: Palette Mode for non 4: 4: 4 Color Format*, document JVET-Q0504 of the 17th JVET Meeting, 2020.

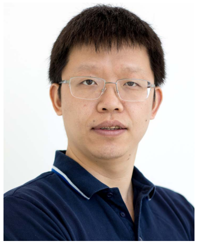
[56] M. Sarwer, Y. Ye, J. Luo, and R.-L. Liao, *CE2-Related: On Maximum Palette Size of VVC*, document JVET-Q0291 of the 17th JVET Meeting, 2020.

[57] X. Xu, Y.-H. Chao, Y.-C. Sun, and J. Xu, *Description of Core Experiment 8 (CE8): Screen Content Coding Tools*, document JVET-N1028 of the 14th JVET Meeting, 2019.

[58] 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 of the 18th JVET Meeting, 2020.

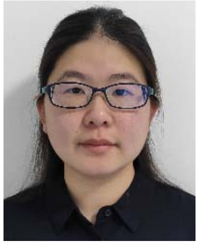
[59] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document VCEG-M33 of the 13th VCEG Meeting, Apr. 2001.

**Tung Nguyen** received the Dipl.-Inf. degree in computer science from the Technical University of Berlin (TUB), Berlin, Germany, in 2008. In 2008, he joined the Fraunhofer Institute for Telecommunications–Heinrich Hertz Institute (HHI), where he has been working as a Student Member since 2004. He is currently with the Video Coding Technologies Group, Video Coding and Analytics Department. His main research interests include efficient compression of image and video that also involves active participation in standardization activities, such as developing the High Efficiency Video Coding (HEVC) and Versatile Video Coding (VVC) standards.



**Xiaozhong Xu** (Member, IEEE) received the B.S. degree in electronics engineering from Tsinghua University, Beijing, China, the M.S. degree in electrical and computer engineering from the Polytechnic School of Engineering, New York University, New York, NY, USA, and the Ph.D. degree in electronics engineering from Tsinghua University. From 2011 to 2013, he worked with Zenverge (acquired by NXP in 2014) on multi-channel video transcoding ASIC design. From 2013 to 2017, he was a Senior Staff Engineer and a Department Manager of multimedia technology development with MediaTek USA Inc. He has been a Principal Researcher and a Senior Manager of multimedia standards with the Tencent Media Laboratory, Palo Alto, CA, USA, since 2017. He also held technical positions with Thomson Corporate Research (now Technicolor) and Mitsubishi Electric Research Laboratories (MERL). He holds more than 100 granted U.S. and global patents. He has been an active participant in video coding standardization activities for more than 15 years. He has successfully contributed to various standards, including AVC and its extensions, AVS1 and AVS3, China, HEVC and its extensions, MPEG-5 EVC, and the most recent VVC standard. He served as a core experiment coordinator and a key technical contributor for screen content coding developments in various standards (HEVC, VVC, EVC, and AVS3). His research interests include multimedia, video and image coding, processing, and transmission. He was a recipient of the Science and Technology Award from the China Association for Standardization in 2020.

**Felix Henry** received the Dipl.-Ing. (M.Sc.) degree in telecommunications and the Ph.D. degree in the domain of wavelet image coding from Telecom ParisTech, Paris, France, in 1993 and 1998, respectively. In 1995, he started his career with the Canon Research Center, France, where he worked on still image compression and video coding. He is currently working as a Video Coding Project Leader with Orange Labs, Issy-les-Moulineaux, France. He participated actively in JPEG2000 standardization. He is a co-inventor of more than 100 patents in the domain of image and video processing. His current research interests include development of HEVC, transform coefficient coding, and high level parallel processing.



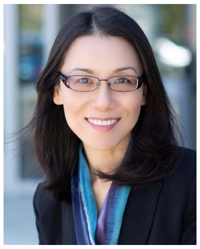
**Ru-Ling Liao** received the B.S. and M.S. degrees in computer science and engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2013 and 2016, respectively. Since 2016, she has been participated in the development of VVC standard. After 2019, she joined Alibaba Inc., Beijing, China, where she is currently a Senior Engineer. Her research interests include video processing and neural network-based video compression technology.

**Mohammed Golam Sarwer** received the M.Phil. degree in electrical engineering from the City University of Hong Kong in 2007 and the Ph.D. degree in electrical engineering from the University of Windsor, Windsor, ON, Canada, in 2011. In 2011, he joined OnMobile Global as a Senior Engineer, for one and half years. From 2015 to 2019, he worked as a Senior Researcher with Sony Electronics and MediaTek Inc. He is currently working with Alibaba Global where he participates VVC and AV2 standardization activities. He has published more than 40 peer-reviewed articles in the areas of video and image compression. His current research interests include multimedia data compression and transmission.

**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 an 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.

**Yung-Hsuan Chao** received the B.S. degree in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2011, and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2017. Since October 2017, she has been with Qualcomm Technologies Inc., San Diego, CA, USA. She has been an active contributor to VVC standardization. Her research interests include image and video processing, and video compression.

**Jizheng Xu** (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Shanghai Jiao Tong University, China. In 2003, he joined Microsoft Research Asia, where he is currently a Research Manager. He is an active contributor to ISO/MPEG and ITU-T video coding standards. He has more than 40 technical proposals adopted by international standards, including H.264/AVC, H.264/AVC scalable extension, High Efficiency Video Coding (HEVC), HEVC range extensions, and HEVC screen content coding extensions. He has authored or coauthored more than 140 refereed journal articles and conference papers. He holds more than 30 U.S. patents granted or pending in image and video coding. His research interests include image and visual signal representation, image/video compression and communication, computer vision, and deep learning. He chaired and co-chaired the Ad Hoc Group of exploration on wavelet video coding in MPEG and various technical ad hoc groups in JCT-VC, such as on screen content coding, on parsing robustness, and on lossless coding. He was a Co-Organizer and the Co-Chair of Special Sessions on scalable video coding, directional transform, and high-quality video coding at various conferences. He served as the Special Session Co-Chair for the IEEE International Conference on Multimedia and Expo 2014 and a Guest Editor for the Special Issue on Screen Content Video Coding and Applications for IEEE Journal on Emerging and Selected Topics in Circuits and Systems. He is an Associate Editor of IEEE Transactions on Circuits and Systems for Video Technology.



**Shan Liu** (Senior Member, IEEE) received the B.Eng. degree in electronic engineering from Tsinghua University and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California. She was formerly the Director of the Media Technology Division, MediaTek USA. She was also with MERL and Sony. She is currently a Tencent Distinguished Scientist and the General Manager of the Tencent Media Laboratory. She has been actively contributing to international standards during the last ten years. She directly contributed to and led development effort of technologies and products which have served several 100 million users. She has more than 100 technical proposals adopted into various standards, such as VVC, HEVC, OMAF, DASH, MMT, and PCC. She holds more than 200 granted U.S. and global patents. She has published more than 100 journal articles and conference papers. Her research interests include audio-visual, high volume, immersive and emerging media compression, intelligence, transport, and systems. She was with the committee of Industrial Relationship of IEEE Signal Processing Society from 2014 to 2015. She served as the VP for the Industrial Relations and Development of Asia–Pacific Signal and Information Processing Association from 2016 to 2017. She was named as the APSIPA Industrial Distinguished Leader in 2018. She has been on the Editorial Board of IEEE Transactions on Circuits and Systems for Video Technology since 2018. She received the Best Associate Editor Award in 2019 and 2020. She served as a Co-Editor for H.265/HEVC SCC and H.266/VVC. She also served or serves as a Guest Editor for a few IEEE Transactions on Circuits and Systems for Video Technology (TCSVT) special issues and special sections. She has been serving as the Vice Chair for IEEE Data Compression Standards Committee since 2019.

**Detlev Marpe** (Fellow, IEEE) received the Dipl.-Math. degree (Hons.) from the Technical University of Berlin (TUB), Germany, and the Dr.-Ing. degree in computer science from the University of Rostock, Germany.

He is currently the Head of the Video Coding and Analytics Department and the Head of the Image and Video Coding Group at Fraunhofer HHI, Berlin. He is also active as a part-time Lecturer at TUB. For nearly 20 years, he has successfully contributed to the standardization activities of ITU-T VCEG, ISO/IEC JPEG, and ISO/IEC MPEG in the area of still image and video coding. During the development of the H.264|MPEG-4 Advanced Video Coding (AVC) standard, he was the chief architect of the CABAC entropy coding scheme as well as one of the main technical and editorial contributors to its Fidelity Range Extensions (FRExt) including the now ubiquitous high profile. He was also one of the key people in designing the basic architecture of scalable video coding (SVC) and multi-view video coding (MVC) as algorithmic and syntactical extensions of H.264|AVC. He made successful contributions to the recent development of the H.265|MPEG-H High Efficiency Video Coding (HEVC) standard, including its range extensions and 3D extensions. He is the author or coauthor of more than 200 publications in the area of video coding and signal processing. He holds numerous internationally issued patents and patent applications in this field. His current research interests include image and video coding, signal processing for communications, computer vision, and information theory.

Dr. Marpe is a member of the German Institute for Standardization (DIN) and the Information Technology Society in the VDE (ITG). For his substantial contributions in the field of video coding, he received numerous awards, including the Nomination for the 2012 German Future Prize, the Karl Heinz Beckurts Award 2011, the 2009 Best Paper Award of the IEEE Circuits and Systems Society, the Joseph von Fraunhofer Prize 2004, and the Best Paper Award of the German Information Technology Society in 2004. As a co-editor and a key contributor of the High Profile of H.264|AVC as well as a member and a key contributor of the Joint Collaborative Team on Video Coding for the development of H.265|HEVC, he was a co-recipient of the 2008 and 2017 Primetime Emmy Engineering Award, respectively. His current citations are available at Google Scholar. He serves as an Associate Editor for the IEEE Transactions on Circuits and Systems for Video Technology.

**Gary J. Sullivan** (Fellow, IEEE) received the B.S. and M.Eng. degrees from the University of Louisville, in 1982 and 1983, respectively, and the Ph.D. degree from the University of California at Los Angeles, Los Angeles, CA, USA, in 1991. He is currently a Video and Image Technology Architect with Microsoft Research. He has been a longstanding chairman/co-chairman of various video and image coding standardization activities in ITU-T VCEG, ISO/IEC MPEG, ISO/IEC JPEG, and in their joint collaborative teams since 1996. He has led the development of the Advanced Video Coding (AVC) standard (ITU-T H.264|ISO/IEC 14496-10), the High Efficiency Video Coding (HEVC) standard (ITU-T H.265|ISO/IEC 23008-2), the Versatile Video Coding (VVC) standard (ITU-T H.266|ISO/IEC 23090-3), and various other projects. At Microsoft, he has been the Originator and a Lead Designer of the DirectX Video Acceleration (DXVA) video decoding feature of the Microsoft Windows operating system. He is also a fellow of SPIE. He received the IEEE Masaru Ibuka Consumer Electronics Award, the IEEE Consumer Electronics Engineering Excellence Award, two IEEE Transactions on Circuits and Systems for Video Technology (TCSVT) best paper awards, and the SMPTE Digital Processing Medal. The team efforts that he has led have been recognized by three Emmy awards.