INVITED PAPER

Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC)

By Benjamin Bross, *Member IEEE*, Jianle Chen, *Senior Member IEEE*, Jens-Rainer Ohm, *Member IEEE*, Gary J. Sullivan, *Fellow IEEE*, and Ye-Kui Wang

ABSTRACT | In the last 17 years, since the finalization of the first version of the now-dominant H.264/Moving Picture Experts Group-4 (MPEG-4) Advanced Video Coding (AVC) standard in 2003, two major new generations of video coding standards have been developed. These include the standards known as High Efficiency Video Coding (HEVC) and Versatile Video Coding (VVC). HEVC was finalized in 2013, repeating the ten-year cycle time set by its predecessor and providing about 50% bit-rate reduction over AVC. The cycle was shortened by three years for the VVC project, which was finalized in July 2020, yet again achieving about a 50% bit-rate reduction over its predecessor (HEVC). This article summarizes these developments in video coding standardization after AVC. It especially focuses on providing an overview of the first version of VVC, including comparisons against HEVC. Besides further advances in hybrid video compression, as in previous development cycles, the broad versatility of the application domain that is highlighted in the title of VVC is explained. Included in VVC is the support for a wide range of applications beyond the typical standard- and high-definition camera-captured content codings, including features to support computer-generated/screen content, high dynamic range content, multilayer and multiview coding, and support for immersive media such as 360° video.

KEYWORDS | Compression; H.265; H.266; High Efficiency Video Coding (HEVC); Joint Video Experts Team (JVET); Moving Picture Experts Group (MPEG); standards; versatile supplemental enhancement information (VSEI); Versatile Video Coding (VVC); video; video coding; Video Coding Experts Group (VCEG); video compression.

I. INTRODUCTION

In 2013, the first version of the High Efficiency Video Coding (HEVC) standard was finalized [1], providing about a 50% bit-rate reduction compared with its predecessor, the H.264/MPEG-4 Advanced Video Coding (AVC) standard [2]. Both standards were jointly developed by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). AVC itself had provided about 50% bit-rate reduction compared with the H.262/MPEG-2 Video standard, which had been produced a decade earlier and was also a joint project of the same organizations [3]–[5]. Now, as of July 2020, VCEG and MPEG have also finalized the Versatile Video Coding (VVC) standard [6], aiming at yet another 50% bit-rate reduction and providing a range of additional

Manuscript received March 9, 2020; revised October 31, 2020; accepted November 29, 2020. This work was supported by Fraunhofer-Gesellschaft. *(Corresponding author: Benjamin Bross.)*

Benjamin Bross is with Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, HHI, 10587 Berlin, Germany (e-mail: benjamin.bross@hhi.fraunhofer.de).

Jianle Chen is with Qualcomm Inc., San Diego, CA 92121 USA.

Jens-Rainer Ohm is with the Institute for Communications Engineering, RWTH Aachen University, 52062 Aachen, Germany.

Gary J. Sullivan is with Microsoft Corporation, Redmond, WA 98052 USA.

Ye-Kui Wang is with Bytedance Inc., San Diego, CA 92130 USA (e-mail: yekui.wang@bytedance.com).

Digital Object Identifier 10.1109/JPROC.2020.3043399

functionalities. The VVC standard is accompanied by an associated metadata specification called the versatile supplemental enhancement information (VSEI) standard [7].

Currently, along with the major increases in the reach and speed of broadband Internet services, the share of the video in global data traffic is already about 80% and is continuing to grow [8]. In addition, the proportion of household TV sets with 4k (3840 × 2160) resolution is steadily growing, and these higher resolution TVs require higher quality video content in order to reach their full potential. Although practically every 4k TV is equipped with an HEVC decoder to play back high-quality 4k video, the data rates necessary to deliver that content are still rather high, stretching the limits of broadband capacity. This illustrates the need for even more efficient compression than the current HEVC standard can provide—a need now further addressed by VVC.

In addition to its high compression performance, VVC was designed to facilitate efficient coding for a wide range of video content and applications, including the following:

1) video beyond the standard and high definitions, including even higher resolution (up to 8k or larger), high dynamic range (HDR), and wide color gamut;
2) computer-generated or screen content, as occurs especially in computer desktop screen sharing and online gaming applications;
3) 360° video for immersive and augmented reality.

Furthermore, the first version of VVC includes flexible mechanisms for resolution adaptivity, region-based access, layered coding scalability, coding of various chroma sampling formats, and flexible bitstream handling, such as the extraction and merging of regions from different coded video bitstreams.

The remainder of this article is organized as follows. Section II lays out the motivation, scope, and common basic hybrid video coding design of the major standards. Section III briefly reviews the HEVC standard and its extensions. The most recent advances in video coding technology, as incorporated in the VVC standard, are described in Section IV. Section V presents coding efficiency results comparing VVC and HEVC to each other and to AVC. Finally, this article is concluded with an outlook in Section VI.

II. VIDEO CODING STANDARDS

Modern video coding standards have been developed to efficiently transmit and store digital video with a variety of requirements on bit rate, picture quality, delay, random accessibility, complexity, and so on. The support for the following applications is of particular importance.

1) Real-time conversational services, for example, video telephony, video conferencing, screen sharing, and cloud gaming, where low delay/latency and reasonable complexity are key requirements (an application recently brought to the forefront by the COVID-19 pandemic);
2) Live broadcast, for example, TV over satellite, cable, and terrestrial transmission channels where the focus is on picture quality, constant or moderately varying channel bit rate, moderate delay, and frequent random access points for channel tuning-in and channel switching;
3) Video on demand, for example, video streaming over Internet protocol (IP) where the picture quality, bit rate, and adaptation to transmission channels matter most;
4) Capture, streaming, and storage by digital cameras, for example, as used in smartphones, drones, actions, security cameras, and professional camera systems.

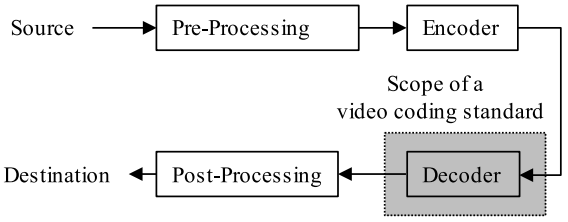

Fig. 1. *Scope of a video coding standard (only the decoder).*

End-to-end video compression technology involves, at the source, an encoder to compress the video into a bitstream and, at the sink, a decoder to decompress the bitstream for consumption. The combination of an encoder and a decoder is commonly referred to as a codec. However, the term is somewhat misleading since encoders and decoders are typically implemented as entirely separate products, and in most applications, the number of encoders is very different from the number of decoders. As depicted in Fig. 1, video coding standards have been specifying only the format of the coded data and the operation of the decoder. This includes the structure and syntax of the bitstream and the processes required to reconstruct the decoded video from it, but not the operations performed by an encoder.

Having the decoder standardized ensures interoperability with all compliant decoder devices while allowing encoders to be designed and operated under application-specific constraints on efficiency, computational complexity, power consumption, latency, and other considerations. For example, in a real-time communication scenario, any particular encoder is unlikely to have the time or computing resources to test all possible coding modes and may, thus, sacrifice some coding efficiency for lower latency and/or complexity. The types and degrees of such algorithmic optimizations are deliberately left outside the scope of the standard.

All video coding standards since H.261 in 1988 [9] have been based on the so-called hybrid video coding principle, which is illustrated in Fig. 2. The term hybrid refers to the combination of two means to reduce redundancy in the video signal, that is, prediction and transform coding with quantization of the prediction residual. Although prediction and transforms reduce redundancy in the video

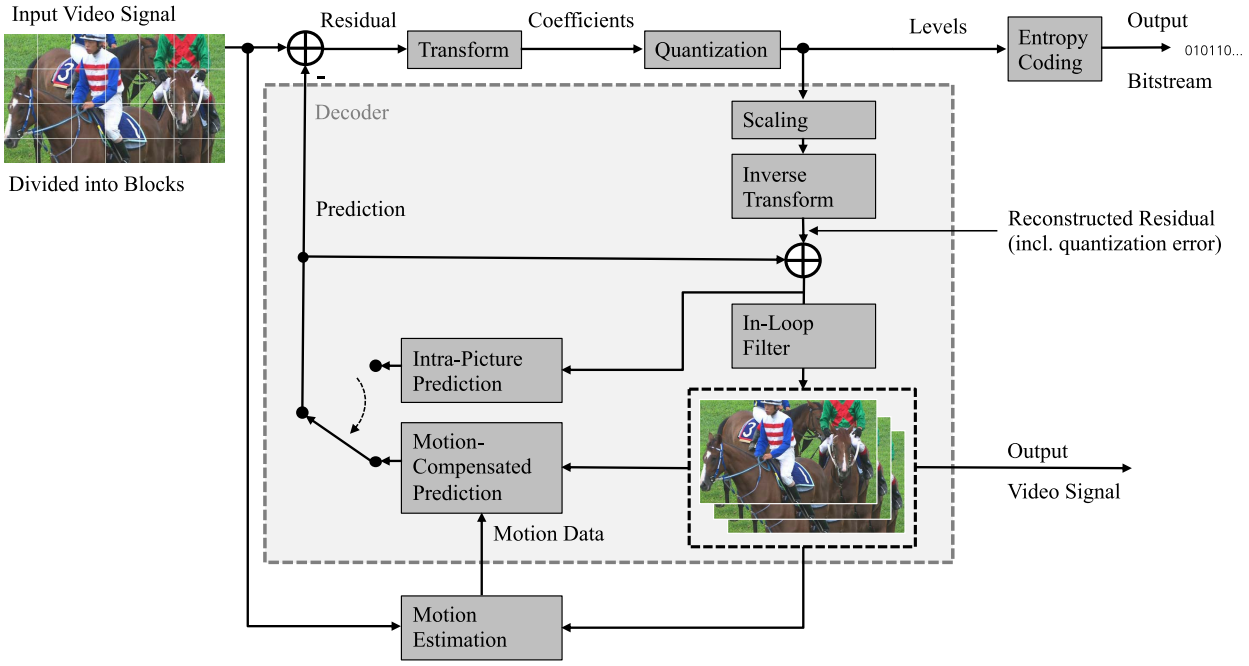

Fig. 2. ***Block diagram of a hybrid video encoder, including the modeling of the decoder within the encoder.***

signal by decorrelation, quantization decreases the data of the transform coefficient representation by reducing their precision, ideally by removing only imperceptible details; in such case, it serves to reduce irrelevance in the data. This hybrid video coding design principle is also used in the two most recent standards HEVC and VVC. For a more detailed review of the previous standards, spanning from H.120 [10] to AVC and also including H.261, MPEG-1 Video [11], H.262/MPEG-2 Video [12], H.263 [13], and MPEG-4 Visual [14], the reader is referred to [3].

Referring to Fig. 2, a modern hybrid video coder can be characterized by the following building blocks.

Block partitioning is used to divide the image into smaller blocks for the operation of the prediction and transform processes. The first hybrid video coding standards used a fixed block size, typically 16×16 samples for the luma prediction regions and 8×8 for the transforms. Starting with H.263, and especially starting with AVC, partitioning became a major part of the design focus. Over the subsequent generations, block partitioning has evolved to become more flexible by adding more and different block sizes and shapes to enable adaptation to the local region statistics. In the prediction stage, this allows an encoder to trade off high accuracy for the prediction (using small blocks) versus a low data rate for the side or prediction information to be signaled (using large blocks). For the coding of residual differences, small blocks enable the coding of fine detail, whereas the large ones can code smooth regions very efficiently. With increasing possibilities for partitioning a picture into blocks, the complexity of an encoder that needs to test the possible combinations and decide which to select also increases compared with a fixed size or limited partitioning set. However, fast partitioning algorithms and advances in computing power have allowed recent standards to provide a high degree of flexibility. AVC, HEVC, and VVC all employ tree-based partitioning structures with multiple depth levels and the blocks as leaf nodes, and VVC additionally provides the ability to use nonrectangular partitions.

Motion-compensated or inter-picture prediction takes advantage of the redundancy that exists between (hence "inter") pictures of a coded video sequence (CVS). A key concept is block-based motion compensation, where the picture is divided into blocks, and for each block, a corresponding area from a previously decoded picture, that is, the reference picture, is used as a prediction for the current block. Assuming that the content of a block moves between pictures with translational motion, the displacement between the current block and the corresponding area in the reference picture is commonly referred to by a 2-D translational motion vector (MV). Finding the best correspondence is typically done at the encoder by a block-matching search that is referred to as motion estimation. The encoder then signals the estimated MV data to the decoder. H.261 used only integer-valued MVs, and this concept of translational motion compensation was later generalized by using fractional-sample MV accuracy with interpolation (with half-sample precision in MPEG-1 and MPEG-2 videos and quarter-sample from MPEG-4 Visual onward), averaging two predictions from one temporally preceding and one succeeding picture (bidirectional prediction in MPEG-1 and MPEG-2 videos) or from multiple

reference pictures with arbitrary relative temporal positions (in standards since AVC). Moreover, the usage of multiple reference pictures from different temporal positions enables hierarchical prediction structures inside a group of pictures (GOP), which further improves coding efficiency. However, when succeeding pictures are used, a structural delay is introduced by requiring a different ordering of the pictures for coding and display [15]. The most recent standard, VVC, even goes beyond the translational motion model by approximating affine motion and using another motion estimation process for motion refinement at the decoder side.

Intra-picture prediction exploits the spatial redundancy that exists within a picture (hence "intra") by deriving the prediction for a block from already coded/decoded, spatially neighboring reference samples. This kind of prediction in the spatial sample domain was introduced with AVC, whereas previous standards used a simplified transform-domain prediction. In AVC, three different types of prediction modes are employed, "DC," planar, and angular, all of them using neighboring samples of previously decoded blocks that are to the left and/or above the block to be predicted. The first, the so-called DC mode, averages the neighboring reference samples and uses this value as a prediction for the entire block, that is, for every sample. The second, that is, the planar mode, models the samples to be predicted as a plane by position-dependent linear combinations of the reference samples. As the third option, the angular modes interpolate the reference samples along a specific direction/angle. For example, the vertical angular mode just copies the above reference samples along each column. HEVC extended these modes, for example, by increasing the number of angles from 8 to 33, whereas the most recent VVC standard not only further extended the number of modes but also incorporates new methods, such as a matrix-based intra-picture prediction (MIP), which was designed using machine learning [16]. Similar to motion information in inter-picture prediction, the encoder signals the estimated prediction information, that is, the intra-picture prediction mode, to the decoder.

Transformation decorrelates a signal by transforming it from the spatial domain to a transformed domain (typically a frequency domain), using a suitable transform basis. Hybrid video coding standards apply a transform to the prediction residual (regardless of whether it comes from inter- or intra-picture prediction), that is, the difference between the prediction and the original input video signal, as shown in Fig. 2. In the transform domain, the essential information typically concentrates into a small number of coefficients. At the decoder, the inverse transform needs to be applied to reconstruct the residual samples. One example of a transform basis is the Karhunen–Loève transform (KLT), which is considered an optimal decorrelator but depends on correlation characteristics of the input signal that are ordinarily not known at the decoder. Another example is the discrete cosine transform (DCT), which has been used since H.261 for hybrid video compression and is also used in the well-known JPEG image compression standard (which was designed around the same time as H.261) [17]. The DCT decorrelates about as well as the KLT for highly-correlated auto-regressive sources and is easier to compute. In later standards starting with H.263 version 3 and AVC, integer-based reduced-complexity transforms are used that are often informally called DCTs although a true DCT uses trigonometric basis functions involving irrational numbers and supports additional factorizations. In order to account for different statistics in the source signal, it can be beneficial to choose between multiple transforms as in HEVC and VVC. Furthermore, applying an additional transform on the transform coefficients as in VVC can further decorrelate the signal.

Quantization aims to reduce the precision of an input value or a set of input values in order to decrease the amount of data needed to represent the values. In hybrid video coding, the quantization is typically applied to individual transformed residual samples, that is, to transform coefficients, resulting in integer coefficient levels. As can be seen in Fig. 2, this process is applied to the encoder. At the decoder, the corresponding process is known as inverse quantization or simply as scaling, which restores the original value range without regaining the precision. The precision loss makes quantization the primary element of the block diagram for hybrid video coding that introduces distortion. Quantization together with scaling can be seen as a rounding operation with a step size controlling the precision. In recent video coding standards, the step size is derived from a so-called quantization parameter (QP) that controls the fidelity and bit rate. A larger step size (larger QP) lowers the bit rate but also deteriorates the quality, which, for example, results in video pictures exhibiting blocking artifacts and blurred details. Typically, each sample is quantized independently, which is referred to as scalar quantization. In contrast to this, vector quantization processes a set of samples jointly, for example, by mapping a block onto a vector from a codebook. At least from the decoder perspective, all recent video coding standards prior to HEVC have employed only scalar quantization. HEVC includes a trick known as sign data hiding that can be viewed as a form of vector quantization, and VVC introduces dependent quantization (DQ), which can be interpreted as a kind of sliding-block vector quantization because the quantization of a sample depends on the states of previous samples. Advanced techniques for optimized encoding with prior standards can also be viewed as vector quantization while appearing to be scalar quantization from the decoder perspective.

Entropy coding assigns codewords to a discrete-valued set of source symbols by taking into account their statistical properties, that is, relative frequency. All recent video coding standards use variable-length coding (VLC) tables that assign shorter codewords to symbols with a higher frequency of occurrence in order to approach the entropy. The way to design codeword tables in earlier standards was based on the Huffman coding (with minor adjustments).

VLC is typically applied to encode and decode the vast majority of the data, including control data, motion data, and coefficient levels. AVC further improved the VLC scheme for coefficient level coding by using a context-adaptive VLC (CAVLC). A context is determined by the value or a combination of values of previous symbols, which can be used to switch to a VLC table designed for that context. Furthermore, AVC was the first video coding standard that introduced context-adaptive binary arithmetic coding (CABAC) as a second, more efficient entropy coding method. CABAC still uses VLC tables to map symbols, such as the coefficient levels to binary strings (codewords). However, the binary strings are not written directly to the bitstream, but, instead, each bit in the binary string is further coded using binary arithmetic coding with context-adaptive probability models. Due to its high efficiency, CABAC has become the sole entropy coding method in the succeeding HEVC and VVC standards.

In-loop filtering is a filtering process (or combination of such processes) that is applied to the reconstructed picture, as illustrated in Fig. 2, where the reconstructed picture is the combination of the reconstructed residual signal (which includes quantization error) and the prediction. The reconstructed picture after in-loop filtering can be stored and used as a reference for inter-picture prediction of subsequent pictures. The name in-loop filtering is motivated by this impact on other pictures inside the hybrid video coding prediction loop. The main purpose of the filtering is to reduce visual artifacts and decrease reconstruction errors. H.263 version 2 is the first standard that used a deblocking in-loop filter, which became a core feature in version 1 of AVC. This filter was designed to be adaptive to the quantization fidelity, so it can attenuate the blocking artifacts introduced by the quantization of block-based prediction residuals while preserving sharp edges in the picture content. HEVC adds a second in-loop filtering stage called sample adaptive offset filtering, which is a nonlinear filter applied after deblocking to attenuate ringing and banding artifacts. In the emerging VVC standard, an adaptive loop filter (ALF) was introduced as a third filter, where, typically, the filter coefficients are determined by minimizing the reconstruction error using a Wiener filter optimization approach. Moreover, in VVC, another process known as luma mapping with chroma scaling (LMCS) can also be applied before the others in the in-loop processing stage.

The next two sections describe the recent developments made over earlier hybrid video coding designs for the HEVC standard and, in more detail, for the most recent VVC standard.

III. HIGH EFFICIENCY VIDEO CODING

The first version of the HEVC standard was finalized in January 2013 and approved as ITU-T H.265 and ISO/IEC 23008-2. At that time, new types of digital video and applications had been emerging. These include picture resolutions beyond HD, such as 4k/UHD, as well as wider color gamut and HDR, both of which require an increased bit depth from 8 to 10 bits per color component sample. At the same time, other formats, such as interlace-scanned video, became less relevant due to advances in display technology (with digital flat panels replacing analog cathode-ray tube displays). While AVC incorporates block-level features optimized for interlaced video, HEVC does not burden decoders with additional complexity for this and, instead, only provides a basic, yet efficient, picture-level method to encode interlaced video using the same set of block-level coding tools as for progressive-scan video.

A. First Version

The first version (v1) of HEVC generalized and improved hybrid video coding beyond the concepts of AVC with a focus on higher resolutions and improved coding efficiency, in general. The following provides an overview of the main features for each part of the hybrid video coding design and a brief description of its high-level picture partitioning and the interfaces to systems and transport layers. For a more detailed description of HEVC and a discussion of its coding efficiency, the reader is referred to [18] and [19].

1) Block Partitioning: As previously mentioned, HEVC introduces a flexible, quadtree-based partitioning scheme that includes larger block sizes. This partitioning scheme is characterized by the following elements.

Coding tree units and quadtree-based block partitioning: In AVC, as well as in previous standards since H.261, a macroblock represents the basic processing unit for further segmentation for the prediction and subsequent transform steps of the hybrid coding scheme. The size of the macroblock, which is the maximum size used in prediction, is fixed to 16×16 luma samples. The color video has three color component planes, so, in addition to the luma samples, the macroblock also has two blocks of chroma samples, which typically have half the width and half the height of the luma block—a sampling format known as the 4:2:0 chroma format. Other, less widely used formats are 4:4:4, in which the chroma planes have the same resolution as luma, 4:2:2, in which the chroma has half the width but the same height as the luma. The monochrome video has only a single component plane and is sometimes called 4:0:0. With increasing picture resolution, homogeneous areas can cover areas larger than this, and the 16×16 size prevents such areas from being coded efficiently. Hence, increasing the maximum block size becomes important for coding higher-resolution video. In HEVC, the macroblock is replaced by the coding tree unit (CTU). The picture area that a CTU covers is selected by the encoder for the entire CVS and can be set to 16×16, 32×32, or 64×64 luma samples. The CTU constitutes the root of a coding quadtree that splits each CTU area recursively into four smaller square areas. The recursive splitting is signaled efficiently by sending a series of binary-valued splitting flags until a leaf node indication or a maximum allowed splitting depth is reached. In HEVC (and VVC), a unit contains blocks of

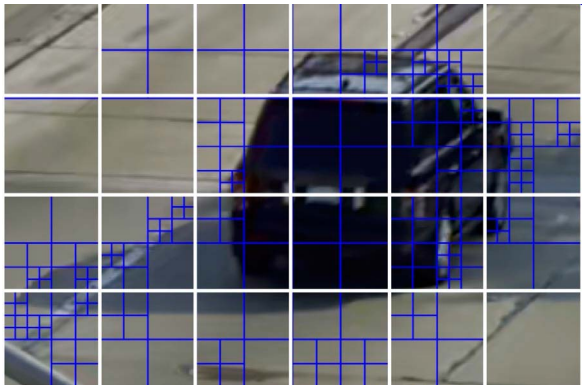

Fig. 3. *HEVC quadtree-based block partitioning with white lines for CTUs and blue lines for CUs.*

samples and syntax to code them. Consequently, a CTU for nonmonochrome video contains three coding tree blocks (CTBs), one for each color component.

Coding unit (CU) is the leaf of the coding quadtree and defines whether the corresponding area of the picture is predicted using inter- or intra-picture prediction. CU block sizes can range in powers of 2 from the maximum CTU area of 64×64 luma samples to the minimum block size of 8×8. An example quadtree partitioning of CTUs into CUs is shown in Fig. 3. It can be seen that flat, homogeneous areas of the picture are covered by large blocks, whereas details and structures with edges are approximated using smaller blocks.

Prediction unit (PU) is the result of a potential further split of a CU for the purpose of having different sets of prediction data, that is, motion information or an intra-picture prediction mode, for different parts of the CU. For CUs coded in an inter-picture prediction mode, eight different splitting modes are defined, as depicted in Fig. 4. This allows motion-compensated prediction with different rectangular shapes, even with narrow ones, that is, when one side is more than twice larger than the other side, which was not possible in AVC. These modes are shown in Fig. 4 and referred to as asymmetric motion partitioning. For intra-picture coded CUs, only a quad split into PUs is allowed. However, intra-picture coded PUs only define the intra-picture prediction mode, whereas the size of the prediction is defined by the transform size as described below.

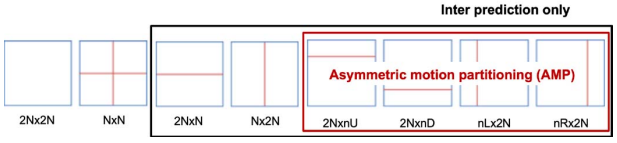

Fig. 4. *The eight different modes in HEVC for partitioning a CU into PUs.*

Transform unit (TU) and residual quadtree transform (RQT) are used to further split a CU for the purpose of transforming the prediction residual using another nested quadtree partitioning with the CU as root and the TUs as leaves. While the most efficient AVC profile (its high profile) defines 4×4 and 8×8 (integerized) DCTs, the RQT in HEVC further allows larger transform sizes for the DCT, that is, 16×16 and 32×32. This additional flexibility enables an encoder to adapt to varying space–frequency characteristics for the DCT. Because each TU has three color components, each TU contains three transform blocks (TBs). TBs in HEVC are always square and have widths and heights that are powers of 2. However, for inter-picture coded CUs, a single TU can span over multiple PUs, for example, two rectangular PUs. This is not allowed in previous standards, such as AVC, where the transform size is always a subset of the prediction size. For intra-picture coded CUs, the prediction is actually performed at the TU level, with the prediction of each TU relying on neighboring samples in another TU that first needs to be reconstructed, and the reconstruction requires performing both the prediction and inverse transform for the neighbor block. Since the splitting of a CU into smaller TUs also increases the correlation between the smaller blocks and the neighboring reference samples, the TU-based prediction process also brings additional coding efficiency for intra-picture prediction.

2) Motion-Compensated or Inter-picture Prediction: Motion-compensated prediction in HEVC, as in AVC, uses a translational motion model with luma MVs in quarter-luma-sample precision, with the ability to reference multiple stored reference pictures using either uniprediction (a motion-compensated prediction generated using one MV and one reference picture) or biprediction (a prediction generated by averaging the predictions from using two MVs and reference pictures in this manner). Beyond this, HEVC includes the following improvements.

Higher quality interpolation filtering is achieved by introducing longer filters and removing an intermediate rounding step. In AVC, quarter-sample values for luma are calculated by applying a six-tap filter to generate the neighboring half-sample value or values, rounding the results to the sample bit depth, and then averaging two neighboring values. For chroma samples, AVC only applies two-tap filtering for all positions. HEVC introduces a consistent separable interpolation process without intermediate rounding for all positions, using an eight-tap filter (specifically, an eight-tap filter for the luma half-sample positions and a seven-tap filter for the quarter-sample positions). For chroma fractional positions, different (four-tap) filters are used.

Improved motion data coding is realized by predicting MV values using a list of predictors and block merging that derives the complete motion information based on neighboring motion data. Typically, the components of an

MV are differentially coded using an MV prediction (MVP) and an MV difference (MVD). In AVC, a single MVP is derived using either median or directional prediction from up to four already coded, spatially neighboring MVs. HEVC replaces the implicit derivation by explicitly signaling one of two potential MVP candidates that are derived from five spatially neighboring and two temporally colocated MVs, where "temporally colocated" refers to MVs used when coding a corresponding location in a particular previously decoded picture. This use of explicit signaling to select among MVP candidates is known as advanced MVP (AMVP). In both AVC and HEVC, MVP-based motion data coding still requires an indication of whether uniprediction or biprediction is applied and, for each MV, an indication of which stored reference picture it refers to. Two reference picture lists (RPLs) are constructed for inter-picture referencing purposes, called list 0 and list 1, where one picture from one list is used for performing uniprediction and one picture from list 0 and one from list 1 are used for biprediction. A reference picture in such a list is selected by an index into the list called a reference picture index. The so-called direct or skip modes in AVC do not signal any motion data; instead, the MVs and reference indices are derived from spatially and temporally neighboring blocks. The skip mode in unipredictive slices derives the list 0 MV from the MVP, and the list 0 reference picture index is 0, referring to the first reference picture in the list. In bipredictive slices, the spatial direct or skip modes derive list 0 and list 1 MVs and reference picture indices from spatially neighboring blocks, whereas the temporal direct or skip modes derive list 0 and list 1 MVs and reference indices from the temporally colocated block. The selection of the skip mode further indicates that the current block does not have a coded residual. HEVC replaces the direct and skip modes by introducing block merging, which derives motion data from one of five merging candidates. The candidates are derived from spatially or temporally neighboring blocks, and only a merge index is signaled to select among the merging candidates. This creates regions of equal motion data, thus enabling us to jointly code regions with equal motion across block boundaries from different quadtree branches. The combination of AMVP and the merge mode is quite effective at establishing a coherent motion representation in the decoded video. The skip mode in HEVC applies block merging without coded residual data.

3) Intra-Picture Prediction: In principle, HEVC intra-picture prediction employs the same types of modes as in AVC, namely DC, planar, and directional angular modes. The more flexible block structures with larger block sizes allow for the following main improvements.

Increased number of angles: From eight angles in AVC to 33 in HEVC for the directional prediction, exploiting the increased number of reference samples available with larger block sizes. The increase comes from adding bottom-left to top-right diagonal directions and using a finer resolution of angles, with a denser coverage around the horizontal and vertical directions. The prediction accuracy is also improved by using bilinear interpolation between the reference sample positions with 1/32 sample precision.

Improved most probable mode (MPM) coding: It is motivated by the increased number of prediction modes. In AVC, the prediction mode can be either signaled using a flag indicating to use the mode inferred from neighbors as the MPM or with a fixed-length code to select among the less probable modes. HEVC extends the MPM concept by constructing a list of three MPMs from the modes of the neighboring blocks to the left and above the current block. An MPM index indicates which MPM is selected, and in case a non-MPM mode is selected, a fixed-length code indicates one of the remaining 31 modes.

4) Transform and Quantization: As mentioned earlier, the introduction of the coding quadtree with nested RQT allows variable power-of-2 transform sizes from 4×4 to 32×32. As in AVC, integer transforms are applied to avoid implementation-dependent precision issues. The 2-D core transforms in HEVC are integer approximations of scaled DCT basis functions, realized by applying 1-D transforms sequentially for rows and columns. The basis functions for all four DCT-based integer transforms have been designed such that they can be extracted from those of the 32-point transform by subsampling. Besides these new DCT-based integer transforms, the following additional transform-related features are introduced in HEVC.

Discrete sine transform (DST) replaces the DCT for prediction residuals resulting from directional intra-picture prediction when the block size is 4×4. It was found that the DST provides better energy compaction in cases where the prediction error increases with increasing distance from one of the block boundaries, which is typically the case for intra-picture prediction due to increasing distance from the reference boundary. Like the DCT, the DST is also simplified and incorporated as a 2-D separable transform. Its bases are integer approximations of scaled DST basis functions. Due to the limited compression benefit for larger block sizes and associated implementation complexity, the DST is restricted to 4×4 luma TBs.

Transform skip is another mode that skips the transform step and, instead, directly quantizes and codes the residual samples in the spatial domain. For certain video signals, such as computer-generated screen content with sharp edges, applying a transform can, sometimes, decrease the coding efficiency. Skipping the transform for such content addresses this issue and can also avoid "ringing" artifacts.

Transform and quantization bypass allows an encoder to skip both the transform and quantization to enable mathematically lossless end-to-end coding. A CU-level flag controls this mode, thereby enabling efficient regionwise lossless coding.

Sign data hiding is used to conditionally skip the signaling of the sign bit of the first nonzero coefficient in a 4×4 subblock. The sign bit is inferred from the parity

of the sum of the coefficient amplitudes when it is not coded. To implement this, the encoder needs to select one coefficient and alter its amplitude in cases where the parity does not indicate the correct sign of the first coefficient.

5) Entropy Coding: The higher coding efficiency of the AVC CABAC entropy coding method compared with CAVLC motivated the decision to have CABAC as the only entropy coding method in HEVC. The basic CABAC design is the same as in AVC, with the following:

1) increased parsing throughput by reducing intersymbol dependencies, especially for parallel-processing hardware architectures;
2) memory reduction by reducing the number of contexts used to store and adapt probability models;
3) improved transform coefficient coding with coefficient scanning and context modeling designed for larger block sizes to increase the coding efficiency.

6) In-Loop Filtering: The in-loop filtering from AVC was kept in HEVC (with a slightly modified deblocking filter), and a nonlinear in-loop filter was added as an additional filtering stage, as follows.

Parallel processing friendly deblocking is enabled in HEVC by aligning the horizontal and vertical block edges, to which the deblocking filter is applied, on an 8×8 grid, in contrast to the 4×4 grid used in AVC. Given the maximum filtering extent of four samples on each side of an edge, each 8×8 block can be filtered in parallel.

Sample adaptive offset (SAO) is introduced in HEVC and consists of two selectable nonlinear filters that are designed to attenuate different artifacts in the reconstructed picture after deblocking. Both filters involve classifying samples and applying amplitude mapping functions that add or subtract offsets to the samples that belong to the same class. The first one is called edge offset that aims to attenuate ringing artifacts. Edge offset classifies each sample into one of five categories (flat area, local minimum, left or right edge, or local maximum) for four gradients (horizontal, vertical, and two diagonals). The second one is called band offset and is designed to attenuate banding artifacts. It subdivides the range of sample values (e.g., 0–255 for 8-bit video) into 32 equally spaced bands. For four consecutive bands, a band-specific offset value is added to each sample inside each of the four bands. The gradient direction for edge offset, the first of the four consecutive bands for band offset, and the four offset values are estimated at the encoder and signaled on a CTU basis.

7) Systems and Transport Interfaces: HEVC inherited the basic systems and transport interface designs from AVC. These include the network abstraction layer (NAL) data unit syntax structuring, the hierarchical syntax relationships, the video usability information (VUI) and supplemental enhancement information (SEI) message mechanisms, and the video buffering model based on a hypothetical reference decoder (HRD). The hierarchical syntax and data unit structures consist of sequence parameter sets (SPSs), multipicture-level picture parameter sets (PPSs), slice-level header syntax, and lower level coded slice data. In the following, the systems and transport interface aspects in HEVC v1 that are essentially different from AVC are briefly summarized. An overview of the AVC designs on these aspects can be found in [3]. More details on the HEVC designs of these aspects can be found in [20]. For simplicity in this description, "HEVC" means HEVC v1, unless otherwise stated.

Random access support: Random access refers to starting the decoding of a bitstream from a picture that is not the first picture in the bitstream in decoding order. To support tuning in and channel switching in broadcast/multicast and multiparty video conferencing, seeking in local playback and streaming, and stream adaptation in streaming, the bitstream needs to include relatively frequent random access points that are typically intra-picture coded pictures but may also be inter-picture coded pictures (e.g., in the case of gradual decoding refresh (GDR) as further discussed in the following).

HEVC includes the signaling of intra random access point (IRAP) pictures in the NAL unit header through NAL unit types. Three types of IRAP pictures are supported, namely instantaneous decoder refresh (IDR), clean random access (CRA), and broken link access (BLA) pictures. IDR pictures constrain the inter-picture prediction structure to not reference any picture before the current GOP and are conventionally referred to as closed-GOP random access points. CRA pictures are less restrictive by allowing certain pictures to reference pictures that precede the current GOP, all of which are discarded in the case of random access. CRA pictures are conventionally referred to as open-GOP random access points. BLA pictures usually originate from splicing together two bitstreams or parts, thereof, at a CRA picture, for example, during stream switching. To enable better systems usage of IRAP pictures, altogether six different NAL unit types are defined to signal the properties of the IRAP pictures, which can be used to enable various types of bitstream access points, such as those defined in the ISO base media file format (ISOBMFF) [21], which are used for random access support in dynamic adaptive streaming over HTTP (DASH) [22].

Video parameter set (VPS): A new type of parameter set, called the VPS, was introduced in HEVC. Although introduced in HEVC v1, the VPS is especially useful to provide a "big picture" of the characteristics of a multilayer bitstream, including what types of operation points are provided, the profile, tier, and level (PTL) of the operation points, layer dependence information, and so on.

Temporal scalability support: HEVC supports temporal scalability (e.g., for extracting lower frame-rate video from a high-frame-rate bitstream) by signaling a temporal ID variable in the NAL unit header and imposing a restriction that pictures of a particular temporal sublayer cannot be used for inter-picture prediction referencing by pictures of a lower temporal sublayer. A subbitstream extraction process is also specified, with a requirement that each

subbitstream extraction output must be a conforming bitstream. Media-aware network elements (MANEs) can use the temporal ID in the NAL unit header for stream adaptation purposes based on temporal scalability.

Profile, tier, and level: In order to restrict the feature set to be supported for specific applications, video coding standards define so-called profiles. HEVC v1 defines the following three profiles: 1) the main profile that is restricted to support only the 4:2:0 chroma format and a bit depth of 8 bits per sample; 2) the Main 10 profile that is based on the main profile with the supported bit depth extended to 10 bits per color component; and 3) the main still picture profile that is also based on the main profile but restricted to have only one picture in a bitstream.

In addition to profiles, HEVC also defines so-called levels and tiers. A level imposes restrictions on the bitstream based on the values of syntax elements and their arithmetic combinations, for example, as combinations of spatial resolution, bit rate, frame rate, and picture buffering capacity. The AVC and HEVC level specifications are generally similar in spirit, with a couple of notable differences: 1) a smaller number of levels is specified in HEVC than in AVC, particularly for the levels with lower picture resolution limits and 2) the highest supported frame rate for operation with picture sizes that are relatively small is 172 frames/s for AVC in most levels, while, for HEVC, this is increased to 300 frame/s. Both of these differences are in response to the general trend of video picture resolutions and frame rates becoming higher as time passes. The concept of tiers was newly introduced in HEVC, mainly to establish high bit rate capabilities for video contribution applications that require higher quality than video distribution applications.

Hypothetical reference decoder: AVC specifies a buffer flow model using picture-based HRD operations with a picture being contained in an access unit (AU) with specified timing. In HEVC, for improved support of ultralow-delay applications, an alternative mode of HRD operation was introduced, which operates on smaller units of data. It specifies a conforming behavior for encoders to send only part of a picture as a decoding unit (DU) with accompanying timing information before the encoding of the remaining areas of the same picture, as well as for decoders to be able to use the timing of DUs to start decoding the received areas before receiving the remaining parts of the picture.

8) High-Level Picture Partitioning: In AVC, the coded macroblocks of a picture are grouped together in slices, each of which can be decoded independent of the other slices in the same picture. When introduced, one of the main purposes of slices was for maximum transfer unit (MTU) size matching for improved channel loss resilience although they could be useful for parallel encoding as well. In HEVC, the basic slice concept was kept, with slices that group together consecutive CTUs in raster-scan order. The more complex slice concepts of flexible macroblock ordering and arbitrary slice ordering have not been widely embraced by industry and, thus, were not carried over from AVC. Instead, new concepts have been introduced to HEVC, which mainly facilitate parallel processing (an important feature given that HEVC is designed for higher-resolution videos).

Tiles represent an alternative, rectangular grouping of CTUs to divide a picture into tile rows and tile columns. The tiles in a picture are processed in raster-scan order, and the CTUs in each tile are processed in raster-scan order within the tile before the CTUs in the next tile are processed. A slice can either contain an integer number of complete tiles such that all the tiles share the same slice header (SH) information, or a tile can contain an integer number of slices with each of these slices being a subset of the tile. The original intent of tiles was enabling parallel encoding and decoding for higher-resolution video [23]. However, with emerging 360° immersive videos, tiles turned out to also be useful for omnidirectional video streaming when used in combination with encoder restrictions and metadata [24]. If an encoder restricts the MVs that it uses to avoid referring to any regions of the reference pictures that are outside of a particular set of tiles, the slices containing these tiles can still be decodable if this set of tiles is extracted from each picture in the bitstream. Such a set is known as a motion-constrained tile set (MCTS). Recent system-level functionalities, especially for immersive videos, have made extensive use of MCTSs.

Wavefront parallel processing (WPP) allows multiple CTU rows to be processed in parallel for decoding (or encoding). When WPP is enabled, the internal state of the CABAC context variables is not carried over to the start of a CTU row from the right-most CTU in the previous row, but rather from the second CTU in the previous row. This allows the decoder (or encoder) to start processing the next row with a two-CTU offset [25]. It should be noted that the WPP term does not appear in the HEVC specification since it is a matter of implementation choice whether the decoder (and/or encoder) actually takes advantage of the feature's parallelism opportunity; in the standard, this is called entropy coding synchronization.

Dependent slice segments have been introduced to provide a separate framing of a coded slice into multiple NAL units. A slice is split into an initial, independent slice segment that contains a full SH and subsequent dependent slice segments that each contain an abbreviated SH [20]. Dependent slice segments are particularly useful for MTU size matching in systems that limit the maximum amount of data in an NAL unit or in combination with WPP, where each CTU row can be packed and transmitted in a dependent slice segment.

B. Extensions

The first version of HEVC was limited to video signals in 4:2:0 chroma format with up to 10 bits per sample and was optimized for consumer-oriented applications with 2-D, single-layer camera-captured content in

the $Y'C_BC_R$ color space. In October 2014, the second version (v2) of HEVC was finalized, in which the format range extensions (REext) add support for more demanding higher quality applications [26], the multilayer extensions for scalability [27], and 3-D multiview video coding [28]. The third version (v3) of HEVC was finalized in February 2015 and added support for combined coding of 3-D video with depth maps [28]. In February 2016, the last major extension, for the coding of screen content material [29], was added in the fourth version (v4). A short summary of these extensions is given in the following.

1) Range Extensions (REext): The main goal of the HEVC range extensions was to extend the 4:2:0 8–10-bit consumer-oriented scope of HEVC v1 by supporting high-quality distribution broadcast (4:2:0, 12 bit), contribution (4:2:2, 10 and 12 bit), production and high-fidelity content acquisition (4:4:4, 16 bit, RGB, high bit rate), medical imaging (4:0:0 monochrome, 12–16 bit, near-lossless), alpha channels and depth maps (4:0:0 monochrome, 8-bit), high-quality still pictures (4:4:4, 8–16 bit, arbitrarily high picture size), and many other applications. The modifications introduced by REext can be divided into the following three categories.

Video format modifications to support chroma formats beyond 4:2:0 and bit depths beyond 10 bits per sample have been kept to a minimum. Here, a rather conservative approach to support the 4:2:2 and 4:4:4 chroma formats without diverging unnecessarily from HEVC v1 was chosen. The modifications include the extension of TB partitioning with existing syntax and transform logic, as well as the adjustment of the intra-picture prediction angles to support the nonsquare rectangular blocks occurring in the 4:2:2 chroma format. For higher bit depths, only the SAO and interpolation precision are extended.

Coding efficiency improvements for extended formats, lossless, and near-lossless coding are achieved by means of modified HEVC v1 tools, as well as by introducing new tools. From HEVC v1, mainly, the transform skip mode was extended to larger block sizes and coupled with a modified residual coding (with a separate CABAC context model and residual rotation). Apart from that, REext includes three new tools to increase coding efficiency: adaptive chroma QP offset allows more flexibility in chroma quantization, cross-component prediction (CCP) exploits remaining statistically redundancies between luma and chroma channels for 4:4:4 video by predicting the chroma spatial residuals from luma using a linear model, and residual differential pulse code modulation (RDPCM) aims to reduce remaining redundancies in the spatial residual signal when the transform is skipped.

Precision and throughput optimizations for very high bit rates and bit depths are achieved mainly by two methods. First, extended precision for the transform coefficients and inverse transform processing enable efficient coding with high bit depths. Second, a modification of CABAC allows to decode multiple coded bits with a single bit-masking and shift operation and can be enabled for increasing the CABAC parsing throughput at very high bit rates.

2) Scalable HEVC Extensions (SHVCs): In HEVC v2, the temporal scalability from v1 is extended by spatial, quality, bit depth, and color gamut scalability, as well as the combinations of these. The scalability is based on a multilayer architecture that relies on multiple single-layer HEVC v1 decoders, that is, it does not modify block-level decoding tools. The reconstruction of a higher enhancement layer from a lower layer, for example, reconstructing UHD from an HD base layer for spatial scalability, is enabled through picture referencing with added interlayer reference picture-processing modules, including texture and motion resampling and color mapping. On the one hand, this allows reusing HEVC v1 decoder cores but, on the other hand, implementing an SHVC-compliant decoder with this architecture increases processing requirements by needing multiple HEVC v1 cores plus the additional modules.

3) Multiview (MV-HEVC) and 3-D Extensions (3-D-HEVC): Based on the same multilayer design introduced in HEVC v2 together with the scalable extension, the multiview and 3-D extensions significantly improve the coding of 3-D video compared with multicast or frame packing with HEVC v1. Similar to the AVC multiview extension, MV-HEVC (in v2 of HEVC), each view of a picture is to be coded in a separate layer with interlayer prediction. 3-D-HEVC (in v3 of HEVC) extends this by coding the view plus its depth map, which allows rendering additional intermediate views. Especially for the depth map coding, statistical dependencies between video texture and depth maps are exploited. This introduces new block-level coding tools, which requires new decoder cores for 3-D-HEVC compared with HEVC v1.

4) Screen Content Coding (SCC) Extensions: Applications such as screen sharing and gaming are mainly based on computer-generated or mixed content. All video coding standards up to HEVC v1 had been mainly designed for camera-captured video, which results in suboptimal exploitation of the different signal characteristics present in screen content. These characteristics are exploited in HEVC SCC (in version 4 of HEVC) by introducing new tools, including intra-picture block copy (IBC), palette mode, adaptive color transform (ACT), and adaptive MV resolution (AMVR). Further detail on these tools is provided in Section IV-B7, as VVC contains a rather similar design for these aspects.

IV. VERSATILE VIDEO CODING

This section describes the most recent standard, VVC, in more detail. It is formally approved as ITU-T H.266 and ISO/IEC 23090-3. The VSEI standard, that is, ITU-T H.274 and ISO/IEC 23002-7, specifying the VUI and some of the SEI messages used with VVC bitstreams,

was developed and approved at the same time [7]. For HEVC and AVC, these aspects are specified directly within the same video coding standard that specifies the coding tools. Apart from achieving major bit-rate savings over its HEVC and AVC predecessors for camera-content video sequences, VVC was designed to provide and improve functionalities and coding efficiency for a broadened range of existing and emerging applications, including:

1) **Video beyond the standard and high definitions** is greatly improved by using more flexible and larger block structures (see Section IV-B1) for higher resolutions and by a luma adaptive deblocking filter designed for HDR video characteristics (see Section IV-B6). Furthermore, profiles that support chroma formats beyond 4:2:0, such as 4:2:2 and 4:4:4, are defined already in the first version of VVC (see Section IV-C8).
2) **Computer-generated or screen content** motivated the inclusion of techniques derived from the HEVC SCC extensions, such as IBC block-level differential pulse code modulation (BDPCM), ACT, palette mode coding, and full-sample adaptive MV precision, as well as an alternative residual coding for transform skip modes (see Section IV-B7).
3) **Ultralow-delay streaming** is facilitated by built-in GDR handling that can avoid bit rate peaks introduced by intra-picture coded pictures and virtual boundaries for improved support of GDR (see Section IV-C1).
4) **Adaptive streaming** with resolution changes benefits from reference picture resampling (RPR) (see Section IV-C6), which allows switching resolutions within a CVS by resampling reference pictures to the picture resolution of the current picture for the purpose of inter-picture prediction.
5) **360° video** for immersive and augmented reality applications is efficiently coded by the motion-compensated prediction that can wrap around picture boundaries, by disabling in-loop filtering across virtual boundaries (see Section IV-B8) and by subpictures with boundary padding (see Section IV-C5).
6) **Multilayer coding** is supported already in the first version of VVC using a complexity-constrained, single-layer-friendly approach that enables temporal, spatial, and quality scalabilities, as well as multiview coding (see Section IV-C7).

In the following, the initial steps toward establishing a new standardization project with compression efficiency beyond HEVC, as well as a short review of the VVC standard development, are covered in Section IV-A. Then, the novel coding tools in VVC that contributes to the overall bit-rate savings are described in Section IV-B. Finally, advances and novelties in the systems and transport interfaces are presented in Section IV-C.

A. Standardization and Development

The development of VVC can be split into two phases, which are summarized in the following. The first phase was the exploration phase, which started in 2015, primarily focusing on investigating the potential for increased coding efficiency without as much consideration of practical complexity constraints. The exploration phase provided evidence that technology with sufficient compression capability beyond HEVC existed, justifying the start of the official standardization phase (the second phase) spanning from 2018 to 2020. This phase targeted to maintain and even increase the coding efficiency while taking implementation and complexity aspects into full consideration and fulfilling a broadened range of application scope.

1) Exploration Phase (2015–2017): The need for even more efficient compression than the current HEVC standard motivated ITU-T VCEG and ISO/IEC MPEG is studying the potential in 2014 and to join forces again in October 2015 for exploring coding technology beyond HEVC in a new team called the Joint Video Exploration Team (JVET). Based, initially, on VCEG key technical area (KTA) software that began being developed in January 2015, by the end of 2017, the JVET had developed the joint exploration model (JEM) software codebase [30], which demonstrated up to 30% bit-rate reduction compared with HEVC.

The coding efficiency improvements achieved in this exploration effort were considered sufficient evidence to issue a formal Joint Call for Proposals (CfP) for new video coding technology in October 2017, and it was agreed that, once the drafting of a formal standard began, the joint team would be renamed to reflect its change of mission, becoming the Joint Video Experts Team, without changing its JVET abbreviation.

2) Standardization Phase (2018–2020): The CfP attracted the submission of proposals from 32 organizations for the coding of three categories of video content: standard dynamic range (SDR), HDR, and 360° video [31]. An independent subjective evaluation conducted in April 2018 showed that all submissions were superior in terms of subjective quality to HEVC in most test cases and that several submissions were superior to the technology previously explored in the JEM framework in a relevant number of cases. Starting with the analysis of the best-performing proposals among all the submissions, the VVC development started in April 2018 with the first draft of the specification document and test model software. After a large number of coding tools had been on the table from the CfP, it was decided to start with a "clean slate" approach. This first draft only included an advanced quadtree with multitype tree (QT+MTT) block partitioning, which was identified as a common element among almost all proposals, and its implementation would heavily affect the design of all other block-based coding tools. On top of that, more coding tools from the

CfP responses and new ones were studied extensively in "core experiments" with regard to coding efficiency and implementation complexity. In cases where a reasonable tradeoff between coding efficiency and complexity was found, additional tools were then adopted to the VVC design.

B. Coding Tools

VVC applies the classic block-based hybrid video coding architecture known from its predecessors. Although the same framework is applied, novel tools are included in each basic building block to further improve the compression.

Table 1 provides an overview of the coding tools in HEVC version 1 and VVC version 1. In the following, the VVC tools will be explained in more detail.

1) Block Partitioning: In VVC, the QT+MTT scheme using quaternary splits followed by binary and ternary splits for its partitioning structure replaces the quadtree with multiple partition unit types that were used in HEVC, that is, it removes the concept of splitting a CU into PUs and TUs and provides a more flexible CU partitioning. Rectangular PU shapes are replaced by rectangular CU shapes resulting from binary and ternary tree splits. The RQT-based TU partitioning is removed as well, and multiple TUs in a CU can only occur from an implicit split of CUs that have a larger size than the maximum transform length and from CUs with intra sub-partitions (see Section IV-B3). Furthermore, the maximum CTU size is increased to 128 × 128 luma samples, and the maximum supported transform length is increased to 64. This tree-based CU partitioning scheme forms the block partitioning structure for VVC, together with sometimes using a separate tree for the chroma components and easing implementation with the concept of virtual pipeline data units, as will be further described in the following.

Coding quadtree with multitype tree: A CTU is first partitioned by a quadtree structure. Then, the quadtree tree leaf nodes can be further partitioned by a multitype tree structure. There are four splitting types in the multitype tree structure: vertical binary splitting, horizontal binary splitting, vertical ternary splitting, and horizontal ternary splitting. The multitype tree leaf nodes are called CUs, and unless the CU is too large for the maximum transform length, this segmentation is used for the prediction and transform processing without any further partitioning. This means that, in most cases, the CU, PU, and TU have the same block size in the QT+MTT coding block structure. Other than when the CU is too large for the maximum transform size, exceptions also occur when intra sub-partitions (see Section IV-B3) or subblock transforms (SBTs) (see Section IV-B4) are employed. This also means that VVC supports nonsquare TBs in addition to square ones. Fig. 5 shows a CTU divided into multiple CUs with a QT+MTT coding block structure, where the solid block edges represent quadtree partitioning and the dotted edges represent multitype tree partitioning with either binary or ternary splits. The size of the CU may be as large as the CTU or as small as 4 × 4 in units of luma samples. The QT+MTT partitioning provides a very flexible block structure to adapt to the local characteristics, as can be seen in the example overlay in Fig. 6. Furthermore, at the leaf node of the multitype tree, there

Table 1 Overview of Coding Tools in HEVC and VVC

HEVC version 1	VVC version 1
Block partitioning	
64×64 max. CTU size	128×128 max. CTU size
Quadtree (QT)	Quadtree plus Multi-Type Tree (QT+MTT)
Coding Units (CU)	Coding Units (CUs)
Prediction Units (PU)	Chroma Separate Tree (CST)
Residual Quadtree Transform (RQT)	Local Dual Tree
	Virtual Pipeline Data Units (VDPUs)
Motion Compensated or Inter-Picture Prediction	
Merge Mode	Extended Merge Mode and MVP with
Advanced MVP	History-based MV Prediction (HMVP)
	Pair-wise Average MV Prediction Candidate
	Subblock-Based Temp. MV Pred. (SBTMVP)
	Merge with MVD (MMVD)
	Symmetric MVD (SMVD)
	Adaptive MV Resolution (AMVR)
8-tap IFs	8-tap Interpolation Filters (IF)
	Geometric Partitioning Mode (GPM)
	Bi-prediction with CU-level Weights (BCW)
	Combined Intra/Inter-picture Prediction (CIIP)
	Decoder-side MV Refinement (DMVR)
	Bi-Directional Optical Flow (BDOF)
	Affine Motion
	Pred. Refinement with Optical Flow (PROF)
Intra-Picture Prediction	
33 Angles	93 Angles
Linear interpolation	4-tap IFs (2 sets of filters)
DC, Planar	DC, Planar
	Position-Dependent Pred. Combination (PDPC)
	Multiple Reference Lines (MRL)
	Matrix-based Intra-picture Prediction (MIP)
	Cross-Component Linear Model (CCLM)
	Intra Sub-Partitions (ISP)
Transforms and Quantization	
Square transforms (max. 32×32)	Non-square transforms (max. 64×64)
	Multiple Transform Selection (MTS)
	Non-Separable Secondary Transform (LFNST)
	Subblock Transform (SBT)
	Adaptive chroma QP offset
Sign Data Hiding	Sign Data Hiding (SDH)
	Dependent Quantization (DQ)
	Joint Coding of Chroma Residuals (JCCR)
Entropy Coding	
CABAC	CABAC with high-accuracy multi-hypothesis probability estimates
Coefficient groups	Additional coefficient group sizes
Reverse diagonal, hor. and ver. coefficient scan	Reverse diagonal coefficient scan only
	Improved probability model selections for absolute transform coefficient levels
In-Loop Filtering	
	Luma Mapping with Chroma Scaling (LMCS)
Deblocking	Deblocking Boundary Handling Modifications
	Deblocking Long Filter
	Luma-Adaptive Deblocking
SAO	Sample Adaptive Offset (SAO)
	Adaptive Loop Filter (ALF)
	Cross-Component ALF (CC-ALF)
Special Modes	
PCM	
4×4 TS	4×4–32×32 Transform Skip (TS)
Trans. Quant. Bypass	
Quantization Bypass	
Screen Content Coding	
	Block-Level Differential PCM (BDPCM)
	Transform-Skip Residual Coding (TSRC)
	Intra-picture Block Copy (IBC)
	Palette Mode
	Adaptive Color Transform (ACT)
360-degree Video Coding	
	MV Wrap-Around
	Virtual Boundaries for in-loop filtering

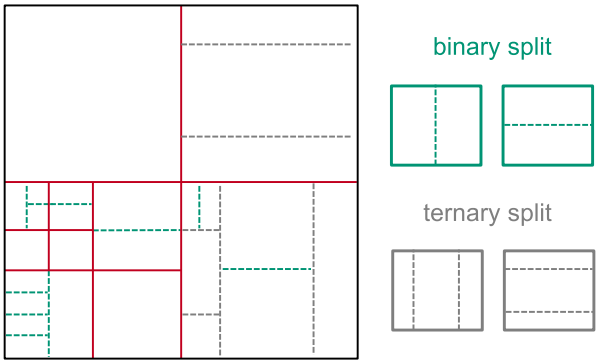

Fig. 5. *Example of quadtree with nested multitype tree coding block structure.*

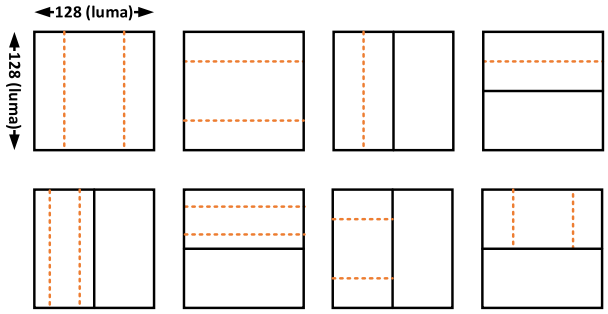

Fig. 7. *Disallowed ternary splitting and binary splitting in VVC when the luma coding block width or height is 128 to enable 64 × 64 VPDU operation.*

is an option to further split a CU into two nonrectangular prediction block partitions in the case of inter-picture prediction, selecting one of 64 geometric partitioning modes (see Section IV-B2).

Chroma separate tree: In VVC, the coding tree scheme supports the ability for luma and chroma to use separate partitioning tree structures. For inter-picture coded slices, the luma and chroma CTBs in one CTU have to share the same coding tree structure. However, for intra-picture coded slices, the luma and chroma can have separate trees. When the separate tree mode is applied, the luma CTB is partitioned into CUs by one QT+MTT structure, and the chroma CTBs are partitioned into CUs by another QT+MTT structure. This means that, when the video is not monochrome, a CU in an intra-picture coded slice may consist of a coding block of the luma component only, coding blocks of two chroma components only, or coding blocks of all three components, whereas a CU in an inter-picture coded slice always consists of coding blocks of all three color components.

Local dual-tree: In typical video encoder and decoder implementations, the average processing throughput drops when many small blocks (more specifically, small intra-picture coded blocks since these need to be decoded sequentially) are present in the coded picture. In the single-coding tree structure, a CU can be as small as 4×4 in units of luma samples, which results in 2×2 chroma coding blocks if the video uses 4:2:0 sampling. To avoid small chroma blocks, a local dual-tree structure is used. With the local dual-tree design, chroma intra-picture coded coding blocks with a size of less than 16 chroma samples or with $2 \times N$ sizes are prevented by using a separate tree locally for the chroma when necessary to prevent such small chroma blocks.

Virtual pipeline data units (VPDUs) are block units in a picture that needs to be held in memory for processing while decoding. In hardware decoders, successive VPDUs can be processed by operating multiple pipeline stages at the same time. The VPDU size would be roughly proportional to the memory buffering size in most pipeline stages, so it is important to keep the VPDU size reasonably small. In the VVC QT+MTT scheme, ternary tree and binary tree splits for CUs with the size of 128×128 luma samples could have led to a VPDU size that was considered too difficult to support. In order to keep the VPDU size at 64×64 luma samples, normative partitioning restrictions (with syntax signaling modification) are applied, disallowing certain splits for CUs with width or height equal to 128, as shown by dashed lines in Fig. 7. The VPDU concept was used to establish these implementation-oriented split restrictions but is not explicitly discussed in the standard.

Fig. 6. *Example of partitioning using the QT+MTT scheme in VVC.*

2) Motion-Compensated or Inter-Picture Prediction: VVC retains and enhances many of the inter-picture prediction features from HEVC, including the two most important motion information coding methods described earlier: AMVP and the merge mode. Furthermore, HEVC's eight-tap high-precision motion compensation

interpolation filter (IF) for luma fractional positions and four-tap IF for chroma fractional positions are also used. On top of these core features, new coding tools are introduced in VVC for increasing the efficiency of inter-picture prediction. VVC introduces subblock-based motion inheritance, in which the current CU is divided into subblocks with equal size (8×8 luma samples) and the MV for each subblock is derived based on temporally colocated blocks in a reference picture. Merge mode with additional MVD coding is added to further enhance the efficiency of the merge mode. A local CU-based affine motion model is used to represent higher-order motion, such as scaling and rotation, where only one set of parameters is coded per CU, while the motion compensation is performed individually per 4×4 subblock using six-tap IFs. VVC also increases the MV precision to 1/16 luma sample in some modes to improve the prediction efficiency for video content with locally varying and nontranslational motion, such as in the case of the affine mode, while HEVC uses only quarter-luma-sample precision. On top of the higher precision MV representations, a block-level AMVR method is applied to customize the balance between the prediction quality and the bit cost overhead for MV signaling. The geometric partitioning mode splits a CU into two nonrectangular partitions to better match motion at object boundaries. The biprediction with CU-level weights (BCW) mode extends simple averaging to allow weighted averaging of the two prediction signals at the block level. To further improve the prediction quality, decoder-side MV refinement (DMVR) and bidirectional optical flow (BDOF) are introduced, which improves the motion compensation without increasing bit overhead. Finally, VVC provides a mode for combining inter-picture and intra-picture prediction to form the final prediction.

For a CU coded in merge mode, a merge candidate list is constructed, and an index is signaled to specify which candidate MVP is used to form the prediction. In VVC, the merge candidate list consists of five types of candidates in the order: 1) MVPs from spatial neighboring CUs; 2) temporal MVP (TMVP) from colocated CUs; 3) history-based MVP from an FIFO table; 4) pairwise average MVP; and 5) zero MVs. The length of the merge list is signaled in SPS, where the maximum allowed length is 6. The way MVs from spatial neighboring CUs and colocated CUs are used is identical to the way that these are handled in the HEVC merge candidate list.

History-based MV prediction (HMVP) provides candidates beyond the local spatial–temporal neighborhood to allow usage of MV information from CUs that are more remote. The HMVP candidates can be used in both merge and AMVP candidate list construction processes. The motion information of previously coded blocks is stored in a table of MVP candidates for the current CU. The table with multiple HMVP candidates is maintained during the encoding/decoding process and is reset (all candidates removed) when a new CTU row is encountered. Whenever there is an inter-picture coded CU, excluding the CUs coded with affine mode, geometric partitioning, or subblock-based TMVP, the associated motion information is added to the table in a first-in-first-out (FIFO) manner. The HMVP table size is 6.

The pairwise average MVP candidate is generated by averaging the MVs of the first two candidates in the existing merge candidate list. The averaged MVs are calculated separately for each RPL. When the merge list is not full after the pairwise average merge candidate is added, zero MVPs are appended at the end until the maximum merge candidate number is encountered.

Subblock-based temporal MVP (SBTMVP): TMVP in merge mode inherits one set of motion information from a temporal colocated CU. The SBTMVP method in VVC allows inheriting the motion information from the colocated picture at a finer granularity, that is, in units of 8×8 subblocks. This requires storing the MVs of the colocated picture on an 8×8 luma sample grid (in contrast to a 16×16 grid in HEVC). SBTMVP attains MVPs for the subblocks within the current CU in two steps. In the first step, the motion displacement to determine the colocated CU is set to the MV of the neighboring CU to the left if it uses the colocated picture as its reference picture. Otherwise, it is set to (0, 0). In the second step, the MVP for each subblock is derived from the MV of its corresponding subblock inside the colocated CU from the first step.

Merge with MVD (MMVD): The VVC merge mode is extended by allowing signaling an MMVD, which only allows a small number of difference values and, therefore, has less bit overhead than AMVP. When one of the first two merge candidates is selected for a CU, an MVD can be signaled to further refine the MV. A set of MVD ranges are predefined, and an index is signaled to indicate how far the final MV can deviate from the predicted MV.

Symmetric MVD (SMVD): When the motion of the current block is on a constant motion trajectory between a temporally past and a temporally future reference picture in display order, corresponding MVs and reference picture indices tend to be symmetrical. SMVD exploits this to save bits for MVDs and reference picture index signaling. When SMVD is applied for a CU, only the MVD for list 0 is signaled. The MVD for list 1 is set to the reverse of the list 0 MVD, and the list 0 and list 1 reference picture indices are implicitly derived at the slice level.

Adaptive MV resolution (AMVR): In inter-picture prediction, MVs with higher resolution, that is, higher fractional sample position accuracy, usually lead to better prediction and, thus, smaller residual energy. However, more bits are required to represent the MVs with higher accuracy. In the HEVC SCC extension, the precision of the MVs is switchable at the slice level between a quarter of a luma sample as in HEVC v1 and integer luma sample precision. The benefit of being able to select integer luma sample precision is clear for SCC (e.g., for computer desktop screen sharing), where the motion in the computer graphics synthesis is often using only integer sample displacements. In such an instance, the integer-only option

avoids wasting bits on sending fractional precision that is not needed. However, to enable a more flexible adaptation for camera-captured video and mixed content and screen content, a CU-level AMVR scheme is supported in VVC. MVDs of a CU with translational motion in AMVP mode can be coded in units of quarter luma samples, half luma samples, integer luma samples, or four luma samples. For the affine AMVP mode, MVDs can be switched among quarter, integer, or 1/16 luma samples. In the case of IBC (see Section IV-B7), the precision of the block displacement vectors can either be an integer or four luma samples. In order to ensure that the final MV (i.e., the sum of the MVP and MVD) uses the same precision as the MVD, the MVP is rounded to the indicated precision. With CU-level switching of MV resolution, a good tradeoff between prediction quality and MV bit overhead can be achieved. The CU-level MV resolution indication is conditionally signaled if the current CU has at least one nonzero MVD component. When half-luma-sample MV accuracy is used in AMVP mode, a six-tap smoothing IF (SIF) is used instead of the eight-tap IF from HEVC.

Geometric Partitioning Mode (GPM) enables motion compensation on nonrectangular partitions of blocks as one variant of the merge mode in VVC. When this mode is used, a CU is split into two partitions by a geometrically located straight line, and two merge indices (one for each partition) are further signaled. In total, 64 different partition layouts are supported by geometric partitioning for each possible CU size from 8×8 to 64×64, excluding 8×64 and 64×8. The location of the splitting line is mathematically derived from the angle and offset parameters of a specific partition. Each part of a geometric partition in the CU is inter-picture predicted using its own motion, and only uniprediction is allowed for each partition, that is, each part has one MV and one reference picture index. The uniprediction motion constraint is applied to ensure that, as in conventional biprediction, only two motion-compensated predictions need to be computed for each CU. After predicting each of the parts, the sample values are combined using a blending processing with adaptive weights along the geometric partition edge.

Biprediction with CU-level weights (BCW): In HEVC, the biprediction signal is generated by averaging two prediction signals obtained from two reference pictures and/or using two MVs. Weighted averaging of the two prediction signals is supported in HEVC but with a somewhat cumbersome scheme that required establishing weights at the slice level and using the reference picture index to control the weight selection. In VVC, this legacy explicit-weighted prediction scheme is kept and extended with CU-level syntax control for weighted averaging. Five weights are allowed in this weighted averaging biprediction, $w \in \{-2, 3, 4, 5, 10\}/8$. For each bipredicted CU, the weight w is determined in one of two ways: 1) for a nonmerge CU, the weight index is signaled after the MVD or 2) for a merge CU, the weight index is inferred from neighboring blocks based on the merge candidate index. BCW is only applied to CUs with 256 or more luma samples (i.e., CU width times CU height is greater than or equal to 256). If all reference pictures are temporally preceding the current picture in display order, for example, for low-delay applications, all five weights are used. Otherwise, only three weights $w \in \{3, 4, 5\}$ are used.

Combined inter-/intra-picture prediction (CIIP): In VVC, when a CU is coded in merge mode, an additional flag is signaled to indicate whether a CIIP mode is applied to the current CU. The CIIP mode can be applied to a CU containing at least 64 luma samples when both the CU width and CU height are less than 128 luma samples. As its name indicates, the CIIP prediction combines an inter-picture prediction signal with an intra-picture prediction signal. The intra-picture prediction signal is generated using the planar mode. The intra-picture and inter-picture prediction signals are combined using weighted averaging, where the weight value is calculated depending on the coding modes of the top and left neighboring blocks.

Decoder-side MV refinement (DMVR) is used to improve the accuracy of the MVs of the merge mode. It searches candidate MVs around the initial MVs in list 0 and list 1 and, like SMVD, is used only with temporally bidirectional prediction. The DMVR searching process consists of an integer sample MV offset search and a fractional sample MV refinement process. The integer sample MV searching calculates the distortion between each pair of candidate reference blocks in list 0 and list 1, and the search range is ± 2 integer luma samples from the initial MVs. The fractional sample refinement is derived by using a parametric error surface approximation instead of using additional searching with distortion measurement comparisons. When the width or height of a CU is larger than 16 luma samples, the CU is split, and DMVR is processed for each 16×16 block separately. The refined MVs are used to generate the inter-picture prediction samples and are also used in TMVP for the coding of subsequent pictures. However, the original MVs are used in the deblocking process and are also used in spatial MVP for subsequent CU coding to ease potential pipelining in hardware implementations.

Bi-directional optical flow before (BDOF) is another technique for improving temporally bidirectional motion representation and is used to refine the biprediction signal of a CU at the 4×4 subblock level. It is applied to CUs coded either in the merge mode or the AMVP mode. Similar to PROF for affine motion, the BDOF refinement is based on the optical flow concept and assumes homogeneous motion of an object within the current CU. For each 4×4 subblock, a motion difference relative to CU MVs is calculated by minimizing the difference between the list 0 and list 1 prediction subblocks using the cross-correlation and autocorrelation of the horizontal and vertical gradients for each prediction sample. The motion difference together with the prediction sample gradients is then used to adjust the bipredicted sample values in the 4×4 subblock.

Affine motion: In HEVC, only a translational motion model is applied in motion-compensated prediction, which

cannot efficiently represent many kinds of motion, for example, zoom in/out, rotation, perspective shifts, and other nontranslational motion effects that often occur in the real-world video. In VVC, a CU-based affine motion mode is introduced to represent nontranslational motion more efficiently. The affine motion model for a CU is described by MVs of two control points located at the top-left and top-right corners (a four-parameter model) or MVs of three control points located at the top-left, top-right, and bottom-left corners (a six-parameter model). In the four- and six-parameter affine AMVP modes, the control-point MVs for the current CU are signaled in the bitstream. Similar to the merge mode for translational motion, the affine merge mode in VVC directly inherits the affine motion model from a neighboring block. In this mode, the control-point MVs of the current CU are derived based on the motion information of the neighboring CUs. To balance the complexity of the motion-compensated prediction of the affine mode against the accuracy of the affine motion representation, the affine motion model is approximated using translational motion for each 4×4 luma subblock, where the translational MV is computed as the displacement of the center of the subblock, calculated according to the affine motion model and rounded to 1/16 sample fractional accuracy. A set of six-tap IFs, instead of eight-tap filters, is used in order to reduce the computational and memory bandwidth complexities. The motion compensation IFs are applied to generate the prediction of the 4×4 luma subblock with the derived MV. The motion compensation for the chroma components also uses 4×4 subblocks. For 4:2:0 video, the MV of a 4×4 chroma subblock is calculated as the average of the MVs of the top-left and bottom-right 4×4 luma subblocks in the corresponding 8×8 luma region.

Prediction refinement with optical flow (PROF): To achieve a finer granularity of motion compensation, PROF can additionally be applied to refine each luma prediction subblock, targeting the effect of samplewise motion compensation. Each prediction sample in a luma subblock is refined by adding a difference derived based on a simplified optical flow equation using the horizontal and vertical gradients of each prediction sample and sample-based MVD relative to the centered subblock MV. PROF is not applied to chroma samples.

3) Intra-Picture Prediction: The samples of an intra-picture coded block are predicted from reference samples in neighboring blocks to the left and above the current block, which has previously been decoded (prior to in-loop filtering) in the same picture. HEVC uses 35 intra-picture prediction modes, including planar, reference sample averaging (also referred to as the DC mode), and 33 directional angular modes. VVC expands the possibilities with tools further described in the following.

93 intra-picture directional prediction angles: For each luma coding block size, VVC offers a set of 65 directional angular modes, plus the DC and planar prediction modes, and employs an "MPM" list with six candidates to efficiently code the selection among the 67 choices. In HEVC, 33 angular prediction directions are defined from 45° to −135° in a clockwise direction. In VVC, the angular precision is basically doubled to produce 65 angles within that same range, and another 28 "wide-angle" prediction modes beyond this angular range can be used for nonsquare blocks. Fig. 8 illustrates an example for an 8×4 ($W \times H$) block where angular prediction modes referencing beyond $2H + 1$ samples from the shorter side to the left (close to 45°) are replaced with wide-angle prediction modes referencing up to $2W + 1$ samples from the longer side above (beyond −135°). There are 14 such selectable wide angles when $W > H$ and another 14 for $H > W$, bringing the total number of wide angles to 28. The replaced modes are signaled using the original mode indices, which are adaptively remapped to the indices of wide angular modes depending on the block size after parsing. The total number of intra-picture prediction modes for any particular block size is constant, that is, 67, and the mode coding method is the same for all block shapes, but the addition of the wide angles for the nonsquare blocks brings the total number of supported directions to 93 and, thus, brings the total number of modes to 95.

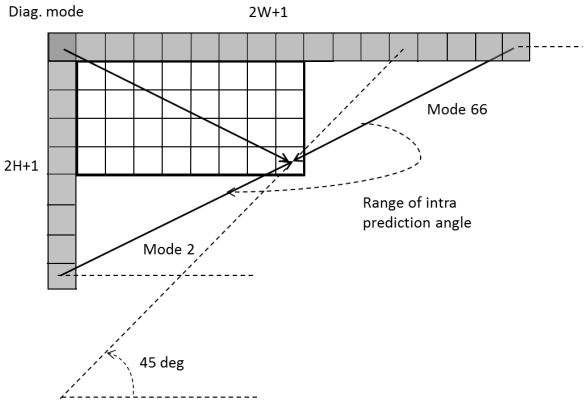

Fig. 8. *Wide-angular intra-picture prediction for an example 8×4 nonsquare block.*

Two sets of four-tap interpolation filters IFs with different frequency cutoffs and 1/32-sample precision are used to generate the prediction samples located at fractional-sample positions for the angular modes. The two sets of four-tap IFs replace lower precision linear interpolation as in HEVC, where one is a DCT-based IF (DCTIF) and the other one is a four-tap SIF. The DCTIF is constructed in the same way as the one used for chroma component motion compensation in both HEVC and VVC. The SIF is obtained by convolving the two-tap linear IF with [1 2 1]/4 filter. The selection of the IF depends on the block size and the angular distance to the horizontal and vertical modes. In general, the sharpening DCTIF is applied more for smaller blocks and for the modes around the

horizontal and vertical directions where the correlation between the reference and original samples tend to be higher. For nonfractional diagonal angles and selected wide angles for blocks with more than 32 samples, luma reference samples are smoothed using a [1 2 1]/4 filter. Compared with HEVC, where an additional strong smoothing can be applied depending on the "flatness" of the reference samples, this is a simplification. Furthermore, reference sample smoothing is applied only to integer-slope modes in luma blocks so that it is not cascaded with interpolation filtering, which is applied to fractional slope modes.

Position-dependent prediction combination (PDPC) further modifies the prediction of the planar, DC, horizontal, vertical, the bottom-left angular mode and its eight adjacent angular modes, and the top-right angular mode and its eight adjacent angular modes. PDPC invokes a combination of prediction with unfiltered boundary reference samples and prediction with filtered boundary reference samples. The final prediction sample is a linear combination of the initial prediction sample and the reference samples with the combination weights dependent on prediction modes and sample location.

Multiple reference line (MRL) prediction uses more reference lines besides the nearest spatial neighboring reconstructed samples for intra-picture prediction. In this mode, instead of using the nearest line of neighboring samples as the reference line for intra-picture prediction, samples from two other lines (a reference line two lines away and a reference line three lines away) can be used.

Matrix-based intra-picture prediction (MIP) is a newly added prediction mode in VVC. It was first proposed as a neural-network-based prediction but was later simplified to use a matrix multiplication and an indexed table of matrices [16]. For predicting the $W \times H$ samples of a rectangular block, MIP performs the following three steps, as shown in Fig. 9: 1) averaging is applied to one left column of H reconstructed neighboring boundary samples and one top line of W reconstructed neighboring boundary samples to get the reduced (downsampled) boundary samples bdry_{red}; 2) a subsequent matrix–vector multiplication with a matrix A_i and an offset vector b_i generates the intermediate prediction signal pred_{red}; and 3) linear interpolation generates the prediction signal pred by upsampling pred_{red}. The matrix coefficients for each MIP mode i are pretrained with 8-bit precision. Overall, 16 16×4 matrices, eight 16×8 matrices, and six 64×7 matrices are specified for MIP.

Cross-component linear model (CCLM) prediction modes are a prediction method specifically for chroma components to exploit cross-component redundancy, in which the chroma samples at positions (x, y) are predicted based on the reconstructed luma samples $\text{rec}_Y(x, y)$ of the same CU by using a linear model $\text{pred}_C(x, y) = \alpha\ \text{rec}_Y(x, y)+\beta$, where the CCLM parameters (α and β) are derived the same way in both the encoder and decoder without explicit signaling. For 4:2:0 video, four neighboring chroma samples at specific locations and their corresponding downsampled luma samples are used in the derivation process, and three CCLM modes are defined based on the locations of the reference samples for the derivation of the model parameters.

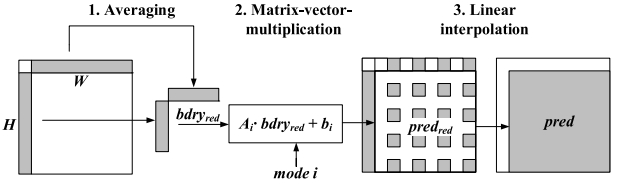

Fig. 9. *MIP process.*

Intra Sub-Partition (ISP) mode divides a luma CU vertically or horizontally into two or four subpartitions depending on the block size. In this mode, all subpartitions share the coding mode information, while the prediction and transform are processed separately. The minimum block size for ISP is 4×8 or 8×4, and the maximum block size is 64×64. If the block size is 4×8 or 8×4, the corresponding block is divided into two subpartitions. Otherwise, it is divided into four subpartitions. Each subpartition corresponds to a TB, with each TB having at least 16 samples.

4) Transforms and Quantization: In HEVC, an integer approximation of the DCT type-II transform is used as the major transform applied to residual signals with square block sizes from 4×4 to 32×32, and as an exception, an integer approximation of the DST type-VII transform is applied for 4×4 intra-picture prediction residual blocks. The conventional uniform reconstruction quantizer design for scalar quantization of the transformed residual can be extended in HEVC by sign data hiding. To achieve better energy compaction of the residual signals and further reduce the quantization error of the transformed coefficients, VVC introduces new tools, which will be reviewed in the following.

Non-square transforms are supported for the non-square TBs in VVC by applying different length transform kernels in horizontal and vertical directions. The maximum transform size is extended to 64×64 to have better energy compaction for the residual signals of large-sized smooth areas.

Multiple transform selection (MTS) is used for residual coding for both inter-picture and intra-picture coded blocks. It provides the ability to select among a predefined subset of (integerized) sinusoidal transforms that include DCT type-II, DST type-VII, and DCT type-VIII transforms for CUs with both width and height smaller than or equal to 32. As shown in Table 2, five combinations of horizontal and vertical transform kernels can be signaled as the (encoder-side) primary transform for a CU. To reduce the complexity of large-size DST type-VII and DCT type-VIII

Table 2 Mapping of MTS Modes to Transform Kernels

MTS Mode Index (Code Word)	Transform Kernel	
	Horizontal	Vertical
0 (0)	DCT-II	
1 (10)	DST-VII	DST-VII
2 (110)	DCT-VIII	DST-VII
3 (1110)	DST-VII	DCT-VIII
4 (1111)	DCT-VIII	DCT-VIII

computation, for blocks with size (width or height, or both width and height) equal to 32, only the coefficients within the 16×16 lower frequency region are retained, and the high-frequency transform coefficients are zeroed out for these transforms. For the TBs with size (width or height, or both width and height) equal to 64, only DCT type-II is used, where only the coefficients within the 32×32 lower frequency region are retained and the high-frequency transform coefficients are zeroed out. In case a low-complexity encoder does not have the resources to test and signal the MTS, an implicit MTS can be used as an alternative. In that case, a combination of DCT type-II and DST type-VII is derived based on the width and the height of the current TB.

Low-frequency non-separable transform (LFNST) can be applied to the low-frequency components of the primary transform to better exploit the directionality characteristics particularly of intra-picture coded CUs with DCT type-II as the primary transform. It is applied between the forward primary transform and quantization at the encoder side and between the inverse quantization scaling and inverse primary transform at the decoder side. In LFNST, a 4×4 or 8×8 nonseparable transform is applied according to the TB size. The 4×4 LFNST is applied to the low-frequency transform coefficients of the TBs with width or height, or both width and height equal to 4, and the 8×8 LFNST is applied for low-frequency transformed coefficients of the TBs with both width and height greater than 4. All transform coefficients outside the 4×4 or 8×8 LFNST zone are discarded (set to zero). To further reduce the computational complexity and storage size of transform matrices, in the case of 8×8 LFNST, only 48 coefficients from the primary transform are used as inputs, and only 16 coefficients are generated as outputs from the secondary transform. Thus, a maximum of 16 coefficients needs to be coded for any TB with LFNST mode enabled. For $4 \times N$, $N \times 4$, and 8×8 blocks, only eight coefficients are output from the secondary transform.

In LFNST, a total of four transform sets and two nonseparable transform matrices (kernels) per transform set are predefined. The transform set to be used is determined based on intra-picture prediction modes. For each transform set, the selected nonseparable secondary transform candidate is further specified by an explicitly signaled LFNST index that is signaled for the CU.

Subblock Transform (SBT) is introduced for inter-picture predicted CUs in VVC. In this transform mode, only a subpart of the residual block is coded. A CU-level flag is signaled to indicate whether the whole residual block or only a subpart of it is coded. In the former case, inter-MTS information is further parsed to determine the transform type of the CU. In the latter case, a part of the residual block is coded with an inferred primary transform type, and the other part of it is zeroed out. The part with coded residual can be one-half or one-quarter the size of the CU and can be located in the left, right, top, or bottom region of the CU, which results in a total of eight SBT modes.

Adaptive chroma QP offset allows extending block-based quantization control for luma, which is similar in spirit as the one introduced in HEVC version 2 by the range extensions. Block-level QP control is widely used in practical implementation for rate control and perceptually optimized encoding approaches. In addition to signal luma QP changes for an area of blocks (quantization groups), chroma QPs are derived from the luma QP of the colocated block via lookup tables. To support a wide range of transfer functions and color formats, the lookup tables are defined by piecewise linear mapping functions that are determined by an encoder and coded in the SPS. Furthermore, VVC extends the range of QP values from 0 to $63 + 6^*(\text{BitDepth}-8)$ in order to achieve low bit rates.

Dependent Quantization (DQ) refers to an approach in which the set of available reconstruction values for a given transform coefficient depends on the reconstruction values that were selected for transform coefficients that precede it in scanning order. The main effect of this approach, in comparison to conventional independent scalar quantization as used in HEVC, is that the average distortion between an input vector given in an M-dimensional vector space (all transform coefficients in a TB) and the closest reconstruction vector can be globally reduced. The approach of dependent scalar quantization in VVC is realized by: 1) defining two scalar quantizers, denoted by Q0 and Q1, with different sets of reconstruction levels and 2) defining a process for switching states between the use of the two scalar quantizers. The location of the available reconstruction levels is uniquely specified by a quantization step size Δ. The scalar quantizer used (Q0 or Q1) is not explicitly signaled in the bitstream. Instead, the quantizer used for a current transform coefficient is determined by the parities (k & 1) of the transform coefficient levels k that precede the current transform coefficient in the scanning order. As shown in Fig. 10, the switching between the two scalar quantizers is realized via a state machine with four states.

Joint coding of chroma residual (JCCR) is used to further reduce the redundancy of the two chroma components' residual signals when they are similar to each other. Instead of signaling the residual for the two chroma components separately, one of three JCCR modes with various weighting combinations of a single-coded chroma residual can be selectively applied at the CU level.

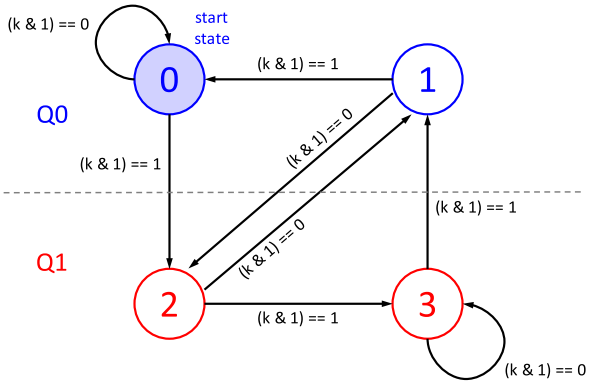

Fig. 10. ***State transition and quantizer selection.***

5) Entropy Coding: As in HEVC, CABAC is used as the single entropy coding method in VVC. The CABAC design in VVC contains various coding efficiency improvements compared with the design in HEVC. The changes in the two main parts of entropy coding, namely the CABAC engine and transform coefficient coding, are further described in this section.

CABAC engine with multihypothesis probability estimate: The CABAC engine in AVC and HEVC uses a table-based probability transition process between 64 different representative probability states. The range representing the state of the coding engine is quantized to a set of four values prior to the calculation of the new interval range. The state transition is implemented using a table containing all the precomputed values to approximate the values of the new probability interval range. In VVC, the basic concept is kept, but the binary arithmetic coder is applied with a multihypothesis probability update model, based on two probability estimates *P0* and *P1* that are associated with each context model and are updated independently with different adaptation rates. The probability estimate *P* that is used for the interval subdivision in the binary arithmetic coder is the average of the estimates from the two hypotheses. The adaptation rates of *P0* and *P1* for each context model are pretrained based on the statistics of the associated binary events.

Improved transform coefficient coding: In HEVC, transform coefficients of a coding block are coded by categorizing them into coefficient groups (CGs or subblocks) such that each CG contains the coefficients of a 4×4 subblock inside a square, power-of-2 sized TB. VVC also adopts the concept of CGs for coefficient coding. Besides the legacy 4×4 CG, additional CG sizes (1×16, 16×1, 2×8, 8×2, 2×4, and 4×2) are introduced due to narrow luma TBs resulting from ISP and small chroma TBs. The CGs inside a TB and the transform coefficients within a CG are coded following a single reverse diagonal scan order. Similar to HEVC, the transform coefficient levels are coded using a combination of different binarizations. This includes truncated unary coding with a cascade of flags that indicate whether the absolute value is greater than 0 (significant), greater than 1, or greater than 2 and Golomb–Rice coding of the remaining absolute values. In VVC, the truncated unary part was modified by adding an additional parity flag to facilitate the state transition for DQ. Compared with HEVC, VVC introduced a more advanced probability model selection for the syntax elements related to absolute values of transform coefficient levels, depending on the values of the absolute levels or partially reconstructed absolute levels in a local neighborhood template. The template comprises two neighboring positions to the right, two below, and one below-right relative to the current scan position.

6) In-Loop Filtering: In VVC, a remapping operation and three in-loop filters can be applied sequentially to the reconstructed picture to modify its representation domain and alleviate different types of artifacts. First, a new sample-based process called LMCS is performed. Then, a deblocking filter is used to reduce blocking artifacts. SAO is then applied to the deblocked picture to attenuate ringing and banding artifacts. Finally, an ALF reduces other potential distortion introduced by the quantization and transform processes. The deblocking filter design is based on the one in HEVC but is extended with longer deblocking filters and a luma-adaptive filtering mode designed specifically for HDR video. While SAO is the same as in HEVC, and the deblocking is very similar, LMCS and ALF are new compared with previous standards. The design of ALF in VVC consists of two operations: 1) ALF with block-based filter adaption for both luma and chroma samples and 2) a cross-component ALF (CC-ALF) for chroma samples.

Luma mapping with chroma scaling (LMCS): Unlike other in-loop filters that, in general, apply filtering processes for a current sample by using the information of its spatial neighboring samples to reduce the coding artifacts, LMCS involves modifying the input signal before encoding by redistributing the amplitudes across the entire representation dynamic range for improved compression efficiency. LMCS has two main components: 1) in-loop mapping of the luma component based on adaptive piecewise linear models and 2) luma-dependent chroma residual scaling for the chroma components. Luma mapping makes use of a forward mapping function and a corresponding inverse mapping function. The forward mapping function is a piecewise linear function with 16 equally sized segments that is signaled in the bitstream. The inverse mapping function does not need to be signaled and is instead derived from the forward mapping function. The luma mapping model is signaled in an adaptation parameter set (APS; see Section IV-C2), and up to four LMCS APSs with different mapping models can be used in a CVS. When LMCS is enabled for a slice, the inverse mapping function is applied to all the reconstructed luma blocks to convert the samples back to the original domain for display output and for storage as reference pictures. For an inter-picture coded block, the forward mapping function needs to be applied to the luma prediction signal

within the decoding process, as the reference pictures are in the original domain. This is not required for intra-picture prediction because the reconstructed signal before inverse mapping is used as a prediction in that case. Chroma residual scaling is designed to compensate for the interaction between the luma signal and its corresponding chroma signals. When luma mapping is enabled, an additional flag is signaled to indicate whether a luma-dependent chroma residual scaling is enabled or not. The chroma residual scaling factor depends on the average value of top and/or left reconstructed neighboring luma samples of the current CU. Once the scaling factor is determined, the forward scaling is applied to both the intra-picture and inter-picture predicted residual at the encoding stage, and the inverse scaling is applied to the reconstructed residual.

Deblocking filter boundary handling modifications: The deblocking filter is applied to the samples adjacent to a CU, TU, and subblock boundary except for the case when the boundary is also a picture boundary, or when deblocking is disabled across slice, tile, or subpicture boundaries (which is an option that can be signaled by the encoder). The deblocking filtering process is applied on a 4×4 grid for CU boundaries and transform subblock boundaries and on an 8×8 grid for prediction subblock boundaries. The prediction subblock boundaries include the PU boundaries introduced by the SBTMVP and affine modes, and the transform subblock boundaries include the TU boundaries introduced by SBT and ISP modes and transforms due to implicit splits of large CUs. As done in HEVC, the processing order of the deblocking filter is defined as horizontal filtering for vertical edges for the entire picture first, followed by vertical filtering for horizontal edges. This specific order enables either multiple horizontal filtering or vertical filtering processes to be applied in parallel threads or can still be implemented on a CTB-by-CTB basis with only a small processing latency.

Deblocking long filters: The deblocking filtering process is similar to that of HEVC. The boundary filter strength (bS) of the deblocking filter is controlled by the values of several syntax elements of the two adjacent blocks, and according to the filter strength and the average QP of the adjacent blocks, two thresholds, tC and β, are determined from predefined tables. For luma samples, one of four cases, no filtering, weak filtering, short strong filtering, and long strong filtering, is chosen based on β and block size. There are three cases: no filtering, normal filtering, and strong filtering for chroma samples. Compared with HEVC, long strong filtering for luma samples and strong filtering for chroma samples are newly introduced in VVC. Long luma strong filtering is used when the samples on either side of a boundary belong to a large block. A sample belonging to a large block is defined as when the width is larger than or equal to 32 for a vertical edge or when the height is larger than or equal to 32 for a horizontal edge. Up to seven samples at one side of a boundary are filtered in the strong filter. Strong chroma filtering is applied when both sides of the chroma edge are greater than or equal to 8 (in units of chroma samples), and three chroma samples from each side are filtered.

Luma-adaptive deblocking further adjusts tC and β of the deblocking filter based on the averaged luma level of the reconstructed samples. When luma-adaptive deblocking is enabled, an offset qpOffset, which is derived based on the average luma level around the filtering boundary, is added to the average QPs of the two adjacent blocks. The value of qpOffset as a function of average luma level is determined by a table of thresholds signaled in the SPS, which may typically be chosen according to the transfer characteristics (the electro-optical transfer function and opto-optical transfer function) of the source video content.

Adaptive loop filter (ALF): Two filter shapes are used in block-based ALF. A 7×7 diamond shape is applied for the luma component, and a 5×5 diamond shape is applied for the chroma components. One among up to 25 filters is selected for each 4×4 block, based on the direction and activity of local gradients. Each 4×4 block in the picture is classified based on directionality and activity. Before filtering each 4×4 block, simple geometric transformations, such as rotation or diagonal and vertical flip, can be applied to the filter coefficients, depending on the gradient values calculated for that block. This is equivalent to applying these transformations to the samples in the filter support region. The idea is to make different blocks to which ALF is applied more similar by aligning their directionality. Block-based classification is not applied to the chroma components.

ALF filter parameters are signaled in an APS. In one APS, up to 25 sets of luma filter coefficients and clipping value indices and up to eight sets of chroma filter coefficients and clipping value indices can be signaled. To reduce bit overhead, filter coefficients of different classifications for the luma component can be merged. In the PH or SH, the IDs of up to seven APSs can be signaled to specify the luma filter sets that are used for the current picture or slice. The filtering process is further controlled at the CTB level. For each luma CTB, a filter set can be chosen among 16 fixed-value filter sets and the filter sets signaled in APSs. For the chroma components, an APS ID is signaled in the PH or SH to indicate the chroma filter sets being used for the current picture or slice. At the CTB level, a filter index is signaled for each chroma CTB if there is more than one chroma filter set in the APS. When ALF is enabled for a CTB, for each sample within the CTB, the diamond-shaped filter selected for the respective 4×4 block is used, with a clipping operation applied to limit the difference between each neighboring sample and the current sample. The clipping operation introduces a nonlinearity by reducing the impact of neighbor sample values that are too different from the current sample value.

Cross-component adaptive loop filter (CC-ALF) can further enhance each chroma component on top of the previously described ALF. The goal of CC-ALF is to use luma sample values to refine each chroma component. This is achieved by applying a diamond-shaped high-pass linear

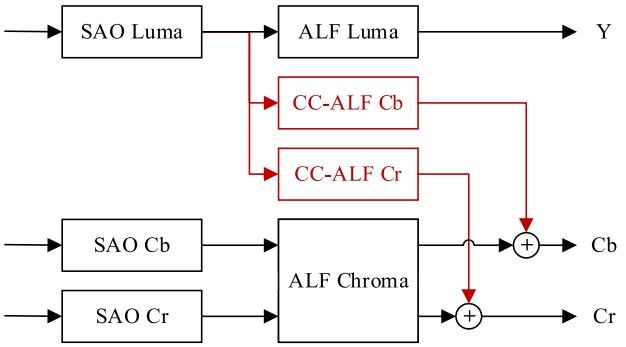

Fig. 11. ***ALF and CC-ALF diagrams.***

filter and then using the output of this filtering operation for chroma refinement. Fig. 11 provides a system-level diagram of the CC-ALF process with respect to the other loop filters. As shown in Fig. 11, CC-ALF uses the same inputs as the luma ALF in order to avoid an additional sequential processing stage of loop-filter processing.

7) Screen Content Coding Tools: One of the design goals for VVC is the efficient coding of computer-generated video content, which exhibits different signal characteristics than camera-captured video. The characteristics mainly include a lack of high-frequency sensor noise, large uniformly flat areas with sharp edges, repeated patterns, highly saturated colors, or a limited number of different colors. Tools to efficiently exploit these characteristics had been added to the HEVC REext and SCC extensions. These tools, with some refinements, have also been used as the basis for the following SCC tools in VVC.

Block-level differential pulse code modulation (BDPCM) is targeting better decorrelation of the screen content prediction residuals by applying samplewise DPCM to the residual instead of a typical frequency transform. Similar to the RDPCM introduced in HEVC REext, the DPCM can be applied in the horizontal (along rows) or vertical direction (along with columns). For intra-picture predicted CUs, the direction is explicitly signaled, and the intra-picture prediction mode is derived from it, for example, vertical DPCM implies vertical intra-picture prediction. However, while the RDPCM in HEVC can be applied to inter-picture prediction residuals, the BDPCM in VVC is restricted to only intra-picture predicted CUs.

Transform skip residual coding (TSRC) adapts the CABAC entropy coding of the spatial transform skip residual block to screen-content-specific characteristics. In the HEVC REext extensions, this statistical difference was already partly considered by using 180° rotation of intra-picture predicted transform skip residuals and a dedicated context model for the flag that indicates an absolute value greater than zero (the significance flag). In VVC, this includes three main aspects: 1) the explicit signaling of the position that indicates the first nonzero value when reverse scanning diagonally from bottom right to top-left is omitted and the scanning direction is inverted (from top-left to bottom-right), as motivated by the higher probability for trailing zeros or insignificant levels at the bottom right corner of the block (due to the lack of energy compaction by a transform); 2) sign indicators can be coded more efficiently using context models due to nonstationarities in the sequence of sign flags even when the global empirical distribution is still almost uniformly distributed; and 3) the binarization of absolute level values is changed resulting in a higher cutoff for the unary binarization prefix, that is, more context-coded "greater than X" flags, and a modified Rice parameter derivation for the Golomb–Rice code suffix. This is motived as well by larger nonstationarities in the empirical distribution of spatial residuals compared with transform coefficients.

Intra-picture block copy (IBC) makes use of repeated patterns inside a picture. It can be seen as a very basic form of motion-compensated prediction with integer MVs (called block vectors) referencing previously coded regions of the same picture instead of previously coded reference pictures. Compared with the HEVC SCC extensions, the IBC in VVC was simplified with regard to the reference sample buffers. In HEVC, IBC relies on the inter-picture design with minor modifications, such as that the RPL only contains the current picture and that a motion or block vector is always in integer precision and has a restriction of the area that it refers to, for example, restricting it to already-decoded samples. However, the IBC in VVC is simplified and decoupled from inter-picture prediction by storing reference samples in a smaller local buffer. This buffer is restricted to contain only the previously coded samples in the current CTU and the CTU to its left. Another difference is having a dedicated IBC merge mode for block vector coding, which is simpler than the VVC inter-picture merge mode. Furthermore, the integer block vector precision from HEVC SCC is extended in VVC to use block-level AMVR as well (see Section IV-B2) but with only full- or four-integer sample precision.

Palette mode is used to represent the sample values in a CU by a set of representative color values. This set is referred to as the palette. For a CU coded in the palette mode, a palette is first signaled, and then, for each sample in the CU, a palette index is signaled. In VVC, for the slices with separate luma/chroma coding trees, the palette is applied on luma (the Y component) and chroma (Cb and Cr components) separately, with the luma palette entries containing only Y values and the chroma palette entries containing both Cb and Cr values. For slices with a single coding tree, palette coding is applied on three color components jointly, that is, each entry in the palette contains Y, Cb, and Cr values. It is also possible to specify a sample that is outside the palette by signaling an escape symbol. For samples within the CU that is coded using the escape mechanism, their quantized values are directly signaled. Although it can be applied to all chroma formats, the palette mode can only be enabled in the profiles that support the 4:4:4 video (see Section IV-C8).

Adaptive Color Transform (ACT) can be applied to reduce the correlation between the three color components

in the 4:4:4 chroma format, which is especially effective for video sequences represented in RGB color spaces. The ACT in VVC is the same as in the HEVC SCC extension. It performs in-loop color-space conversion in the prediction residual domain by adaptively converting the residuals from the input color space (presumed to be RGB) to the YCgCo-R luma–chroma color representation [32]. A flag at the CU level is used to indicate whether the residuals of the CU are coded with the YCgCo-R transformation or in the original color space. The YCgCo-R transformation is fully reversible, so it can even be applied for lossless coding. In order to reduce cache storage requirements, when ACT is enabled for a CVS, the maximum transform size cannot exceed 32×32 samples since ACT requires temporarily storing all three TBs.

8) 360° Video Coding Tools: Another design goal for VVC is the efficient coding of immersive video. This includes 360° video, which is typically coded by representing a 2-D picture that has been generated by a projection mapping from a 3-D sphere. One example of such a mapping is the equirectangular projection format (ERP), in which the sphere is projected onto a rectangular picture with some geometric distortions, especially at the poles. Another mapping is the cube map projection (CMP), where the sphere is mapped onto the six faces of a cube, which are then packed together into one picture. The ability to indicate such formats and the following two techniques have been added to VVC to increase the coding efficiency for video pictures using these projection formats:

MV wrap-around allows for prediction samples to "wrap-around" from the opposite left or right boundary in cases where an MV points outside of the coded area. In ERP pictures, the content tends to be continuous across such a wrap-around due to the 360° nature of the projection mapping, which can result in having a moving object that is partly at the left boundary and partly at the right boundary of a picture.

Virtual boundaries for in-loop filtering prevents applying in-loop filtering across certain "virtual" boundaries, for example, not slice or tile boundaries but corresponding to the CMP face boundaries in CMP pictures. The locations of these boundaries are typically signaled at the CVS level.

C. Systems and Transport Interfaces

VVC inherited many aspects of the systems and transport interfaces from HEVC and the associated header syntax. The bitstream structure is the same as in HEVC except that the concept of an elementary stream is not included. The NAL unit syntax and NAL unit header are both similar as in HEVC, with a small difference in the NAL unit header syntax, where HEVC uses six bits for the NAL unit type field, while VVC uses only five bits, thus allowing half of the maximum number of specified NAL unit types. The VPS, SPS, PPS, and SH followed the same design principle as in HEVC and contain similar types of header parameters. The support of temporal scalability in VVC is also basically the same as in HEVC. Other aspects of the systems and transport interfaces in VVC are summarized in the following, focusing on the differences compared with HEVC.

1) Random Access Support: VVC supports three types of IRAP pictures, two types of IDR pictures (one type with and one type without associated with other pictures that precede them in display order), and one type of CRA picture. These are basically the same as in HEVC. The BLA picture types in HEVC are not included in VVC, mainly because: 1) the basic functionality of BLA pictures can be realized using CRA pictures and an end of sequence NAL unit, the presence of which indicates that the next picture starts a new CVS in a single-layer bitstream and 2) there was a desire for specifying fewer NAL unit types than in HEVC to simplify the design understanding, as reflected by the use of five instead of six bits for the NAL unit type field in the NAL unit header.

Another key difference in random access support between VVC and HEVC is the support of GDR in a more normative manner in VVC. In GDR, the decoding of a bitstream can start from an inter-picture coded picture, and although, in the beginning, some parts of the picture region cannot be correctly decoded, after decoding a number of additional pictures, the entire picture region would become correct for decoding later pictures in the bitstream. (AVC and HEVC can also support a form of GDR, using a recovery point indication SEI message for signaling the GDR random access points and the recovery points.) In VVC, a new NAL unit type is specified for an indication of GDR pictures, and the recovery point is signaled in the picture header (PH) syntax structure, and a bitstream or a CVS within a bitstream is allowed to start with a GDR picture. This means that it is allowed for an entire bitstream to contain only inter-picture coded pictures without a single intra-picture coded picture. The main benefit of specifying GDR support in this way is to provide a conforming behavior for GDR operation. GDR enables encoders to smooth out the bit rate of a bitstream by distributing intra-picture coded slices or blocks across multiple pictures that also contain inter-picture predicted slices or blocks, as opposed to intra-picture coding of entire pictures, thus allowing significant end-to-end delay reduction to improve behavior for ultralow-delay applications, such as wireless display, online gaming, and drone-based applications.

Another GDR-related feature in VVC is the virtual boundary signaling discussed earlier. The boundary between the refreshed region (i.e., the correctly decoded region) and the unrefreshed region at a picture between a GDR picture and its recovery point can be signaled as a virtual boundary, and when signaled, in-loop filtering across the boundary would not be applied; thus, a decoding mismatch for some samples at or near the boundary would

not occur. This can also be useful when the application involves displaying the correctly decoded regions during the GDR process.

2) Adaptation Parameter Set: VVC introduced a new type of parameter set called the APS. An APS conveys picture- and/or slice-level information that may be shared by multiple slices of a picture and/or by slices of different pictures but can change frequently from picture-to-picture with the total number of variants potentially being high and thus not suitable for inclusion into the PPS. Three types of parameters are included in APSs: ALF parameters, LMCS parameters, and scaling list parameters for frequency-specific inverse quantization scaling. The main purpose of introducing APSs is to save signaling overhead.

3) Picture Header: VVC also uses a PH, which contains header parameters for a particular picture. Each picture must have exactly one PH. The PH basically carries those parameters that would have been in the SH if the PH were not introduced but would have the same value for all slices of a picture. These include IRAP/GDR picture indications, flags indicating whether inter-picture and intra-picture coded slices are allowed, picture ordering position syntax, information on RPLs, deblocking, SAO, ALF, QP selection, weighted prediction control, coding block partitioning information, virtual boundaries, colocated picture information, and so on. It often occurs that each picture in an entire sequence of pictures contains only one slice. To avoid needing to have at least two NAL units for each picture, the PH syntax structure can be included either in the PH NAL unit or in the SH in this case. The main purpose of introducing the PH was for saving signaling overhead for cases where pictures are split into multiple slices.

4) Reference Picture Management: Reference picture management is core functionality that is necessary for any video coding scheme that uses multipicture buffering with generalized inter-picture prediction. It manages the storage and removal of reference pictures into and from a decoded picture buffer (DPB) and puts reference pictures in their proper order in the RPLs. Reference picture management in VVC is more similar to HEVC than AVC but is somewhat simpler and more robust. As in those standards, two RPLs, called list 0 and list 1, are derived, but they are not based on the reference picture set concept used in HEVC or the automatic sliding window process used in AVC; instead, they are signaled more directly. Reference pictures are listed for the RPLs as either active or inactive entries, and only the active entries may be used as reference indices for inter-picture prediction of CTUs of the current picture. Inactive entries indicate other pictures to be held in the DPB for potential referencing by other pictures that arrive later in the bitstream.

5) High-Level Picture Partitioning: VVC also includes four different high-level picture partitioning schemes but not the same set as in HEVC. VVC inherited the tiles and WPP from HEVC, with some minor-to-moderate differences. The basic concept of slices was kept in VVC but designed in an essentially different form. VVC introduces subpictures that provide the same region extraction functionality as MCTSs but are designed in a different way to have better coding efficiency and to be friendlier for usage in application systems. More detail about these differences is described in the following.

Tiles and WPP: As in HEVC, a picture can be split into tile rows and tile columns in VVC, intra-picture prediction across tile boundaries is disallowed, and so on. However, the syntax for signaling the tile partitioning has been simplified, by using a unified syntax design for both the uniform and the nonuniform use cases. The WPP design in VVC has two differences compared with HEVC: 1) the CTU row delay is reduced from two CTUs to one CTU and 2) the signaling of entry point offsets for WPP in the SH is optional in VVC, while it is mandatory in HEVC.

Slices: In VVC, the support of conventional slices based on CTUs (as in HEVC) or macroblocks (as in AVC), that is, such that each slice consists of an arbitrary number of CTUs or macroblocks in raster scan order within a tile or within a picture, has been removed. The main reasoning behind this architectural change is as follows. The advances in video coding since 2003 (the publication year of AVC v1) have been such that slice-based error concealment has become practically impossible due to the ever-increasing number and efficiency of intra-picture and inter-picture prediction mechanisms. An error-concealed picture is the decoding result of a transmitted coded picture for which there has been some data loss (e.g., loss of some slices) of the coded picture or a reference picture so that at least some part of the decoded picture is not error-free (e.g., because one or more reference pictures were lost or were error-concealed pictures). For example, when one of the multiple slices of a picture is lost, it may be error-concealed using interpolation of the neighboring slices. While AVC prediction mechanisms provide significantly higher coding efficiency, they also make it harder for algorithms to estimate the quality of an error-concealed picture, which was already a hard problem with the use of simpler prediction mechanisms. Advanced intra-picture prediction mechanisms also function much less well if a picture is split into multiple slices. Furthermore, network conditions have become significantly better in the meantime. As a result, very few implementations have recently used slices for MTU size matching. Instead, substantially, all applications where low-delay error/loss resilience is required (e.g., video telephony and video conferencing) have come to rely on system/transport-level error resilience (e.g., retransmission and forward error correction) and/or picture-based resilience tools (feedback-based resilience, insertion of IRAPs, scalability with uneven protection of the base layer, and so on). With all these, it is very rare that a picture that cannot be correctly decoded is passed to the decoder, and when such a rare case occurs, the system can afford to wait for an error-free picture to be decoded and available

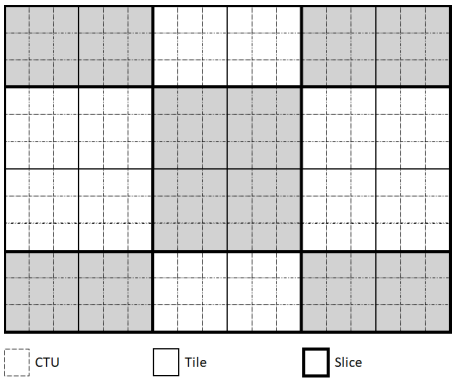

Fig. 12. *Picture with 18 × 12 luma CTUs that are partitioned into 24 tiles and nine rectangular slices.*

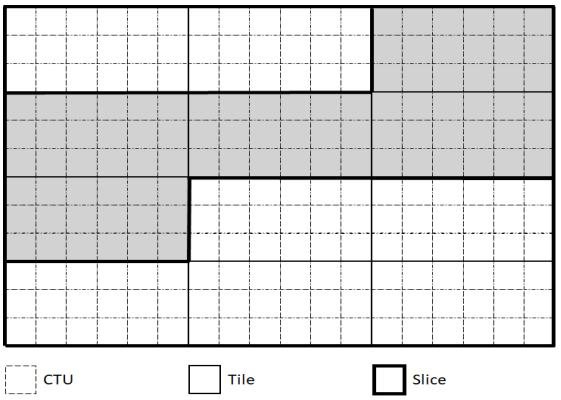

Fig. 14. *Picture with 18 × 12 luma CTUs that are partitioned into 12 tiles and three raster-scan slices.*

for display without frequent and long periods of picture freezing.

Slices in VVC have two modes: rectangular slices and raster-scan slices. As the name implies, rectangular slices always have a rectangular shape, typically consisting of a number of complete tiles that collectively cover a rectangular region of the picture, as shown in Fig. 12. However, it is also possible that a rectangular slice is a subset of a tile and consists of one or more consecutive, complete CTU rows within a tile, as shown in Fig. 13. A raster-scan slice consists of one or more complete tiles in tile raster scan order, and hence, the region covered by a raster-scan slice is typically not a rectangle (e.g., as shown in Fig. 14) although it may also happen to be a rectangle.

The layout of rectangular slices (including the position and the size of each of the slices) is signaled in the PPS based on the layout of tiles. Information on which tiles are included in a raster-scan slice is signaled in the SH.

Subpictures: As mentioned earlier, the subpictures' feature was newly introduced during the development of VVC. Each subpicture consists of one or more complete rectangular slices that collectively cover a rectangular region of the picture, as shown in Fig. 15. A subpicture may be either specified to be extractable (i.e., independent coded of other subpictures of the same picture and of other subpictures of earlier pictures in decoding order) or not extractable. Furthermore, the encoder can control whether in-loop filtering (including deblocking, SAO, and ALF) across the subpicture boundaries is enabled individually for each subpicture.

Functionally, subpictures are the same as the MCTSs that have been supported with SEI messages in HEVC. They both allow independent coding and extraction of a rectangular subset of a sequence of coded pictures, for use cases such as viewport-dependent 360° video streaming optimization and region-of-interest (ROI) applications.

In streaming of 360° video, also known as omnidirectional video, at any particular moment, only a subset (i.e., the current viewport) of the entire omnidirectional video sphere would be rendered to the user, while the user can turn their head at any time to change their viewing orientation and, consequently, the current viewport. While

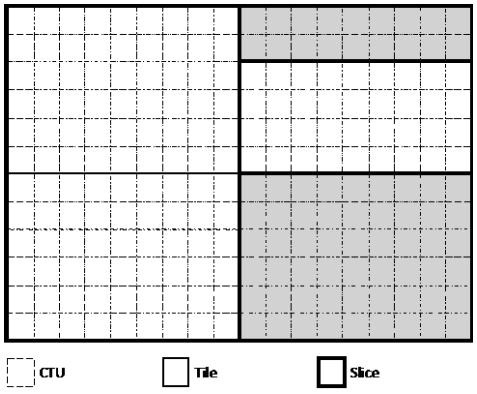

Fig. 13. *Picture partitioned into four tiles and four rectangular slices (note that the top-right tile is split into two rectangular slices).*

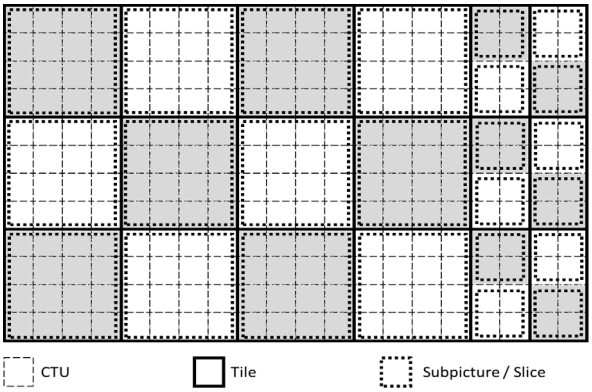

Fig. 15. *Picture partitioned into 18 tiles, 24 slices, and 24 subpictures.*

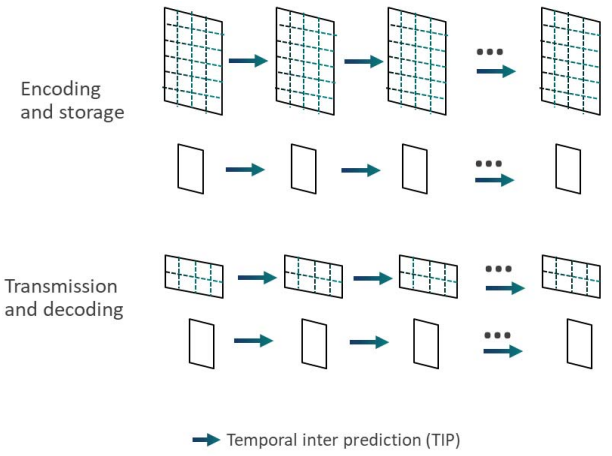

Fig. 16. ***Subpicture-based viewport-dependent 360° video delivery scheme.***

it is desirable to have at least some lower quality representation of the area not covered by the current viewport available at the client and ready to be rendered to the user in case they suddenly change their viewing orientation to somewhere else on the sphere, a high-quality representation of the omnidirectional video is only needed for the current viewport that is actively being rendered to the user. Splitting the high-quality representation of the entire omnidirectional video into subpictures at an appropriate granularity can enable such an optimization.

An example subpicture-based viewport-dependent 360° video delivery scheme is shown in Fig. 16, wherein a higher resolution representation of the full video scene consists of subpictures, while a lower resolution representation of the full video scene does not use subpictures and can be coded with less-frequent random access points than the higher resolution representation. The client receives the full video in the lower resolution, and for the higher resolution video, it only receives and decodes the subpictures that cover the current viewport.

One key difference between VVC subpictures and MCTSs is that the subpicture feature in VVC allows the MVs of a coding block to point outside of the subpicture even when the subpicture is extractable, relying on decoder padding at subpicture boundaries in this case, similarly as at picture boundaries. This allows higher coding efficiency compared with the tight encoder-side motion constraints applied for MCTSs. Another important aspect in the VVC design is that rewriting of the SHs (and PH NAL units, when present) is not needed when extracting one or more VVC subpictures from a sequence of pictures to create a subbitstream that is a conforming bitstream. In subbitstream extraction based on HEVC MCTSs, rewriting of SHs is needed. Although rewriting of SPSs and PPSs is needed in both extraction cases, the number of SPSs and PPSs in a bitstream is low, while each picture has at least one slice and the amount of data in the slices can be very large; therefore, the rewriting of SHs can be a significant burden for application systems. Furthermore, VVC specifies HRD and level definitions for subpicture sequences; thus, the conformance of the subbitstream of each extractable subpicture sequence can be relied upon for system functionalities, such as subpicture-based bitstream extraction and merging.

The layout of subpictures in VVC is signaled in the SPS, and thus, it is constant within a CVS. The trick that enables the extraction of subpicture sequences without rewriting SHs and PHs is through the signaling of subpicture IDs. The subpicture ID of a subpicture can be different from the value of the subpicture index, and the subpicture ID mapping (a list of subpicture IDs, one for each subpicture) is signaled, which may either be constant within a CVS (in which case it is signaled in the SPS) or allowed to change in the pictures within a CVS (in which case it is signaled in the PPS). In the SH, the subpicture ID of the subpicture containing the slice is signaled, and the subpicture-level slice index is also signaled. The subpicture ID and the subpicture-level slice index together tell the decoder where to place the decoded tiles or in-tile CTU rows in the slice into the decoded picture. In an extracted subbitstream containing a subset of the subpictures in each picture of an original bitstream, the same subpicture ID value would still be signaled in the rewritten SPS or PPS, even when the subpicture now has a different subpicture index value. Therefore, even when the raster-scan CTU address of the first CTU in a slice in the subpicture has changed compared with the value in the original bitstream, the unchanged subpicture ID and subpicture-level slice index in the SH can still correctly determine the position of each CTU in the decoded picture of the extracted bitstream.

6) Picture Resolution Changes With Inter-Picture Prediction: In AVC and HEVC, the spatial resolution of pictures cannot change unless a new CVS is started using a new SPS and an IRAP picture. VVC enables picture resolution changes within a CVS without encoding an IRAP picture, thus allowing inter-picture prediction with references to pictures having a different resolution than the current picture that is being decoded. This feature is often referred to as RPR, as it requires resampling of the reference pictures that are used for inter-picture prediction when they have a different resolution than that of the current picture.

The scaling ratio for RPR is restricted to be larger than or equal to 1/2 (factor-of-2 downsampling from the reference picture to the current picture) and less than or equal to 8 (factor-of-8 upsampling). Three sets of resampling filters with different frequency cutoffs are specified to handle various scaling ratios between a reference picture and the current picture. The three sets of resampling filters are applied for the scaling ratios ranging from 1/2 to 1/1.75, 1/1.75 to 1/1.25, and 1/1.25 to 8, respectively. Each set of resampling filters has 16 phases for luma and 32 phases for chroma, which are the same as the number of phases for the filters used for motion compensation

interpolation. In fact, the conventional motion compensation interpolation process is a special case of the resampling process with the scaling ratio in the range from 1/1.25 to 8. The horizontal and vertical scaling ratios are derived based on picture width and height, and left, right, top, and bottom scaling offsets are specified for the reference picture and the current picture.

7) Scalability Support: Due to the support of RPR, in VVC, the support of a bitstream containing multiple spatial scalability layers, for example, two layers with SD and HD resolution, does not require any additional signal processing coding tools, as the upsampling process needed for spatial scalability support can just use the RPR upsampling filter. Nevertheless, some high-level syntax changes (compared with not supporting scalability) are needed for scalability support.

Scalability support is specified in VVC v1, and compared with the scalability support methods in the earlier video coding standards, including in extensions of AVC and HEVC, the design of VVC scalability has been made friendlier to single-layer decoder designs. The decoding capability for multilayer bitstreams is specified in a manner as if there was only a single layer in the bitstream. For example, the decoding capability, such as DPB size, is specified in a manner that is independent of the number of layers in the bitstream to be decoded. Basically, a decoder designed for single-layer bitstreams does not need much modification to be able to decode multilayer bitstreams. Compared with the designs of multilayer extensions of AVC and HEVC, the HLS aspects have been significantly simplified at the sacrifice of some flexibility. For example, an IRAP AU is required to contain a picture for each of the layers present in the CVS.

With the scalability support, not only conventional spatial scalability, quality scalability, and multiview scalability are enabled but also some combinations of scalability and subpictures are enabled in VVC v1. For example, the subpicture-based viewport-dependent 360° video delivery scheme shown in Fig. 16 can be improved by allowing interlayer prediction, as shown in Fig. 17.

8) Profile, Tier, and Level (PTL) Aspects: Two new aspects regarding PTL in VVC have been introduced: the general constraints and the subprofile concept. In HEVC, the PTL syntax structure includes a few general constraint fields for indications, such as whether the bitstream may contain interlaced source content. In VVC, almost every substantial tool or feature has a corresponding general constraint flag. The main reason for this is to enable third parties, for example, an application system standards body or even a company, to be able to easily indicate that certain tools are not used in the bitstream, in case these tools are not conveniently useable by them, without the need of going through the time-consuming process and difficult consensus negotiations that are required for specifying a new VVC profile. The subprofile concept was introduced for a similar purpose. This enables a third party to define

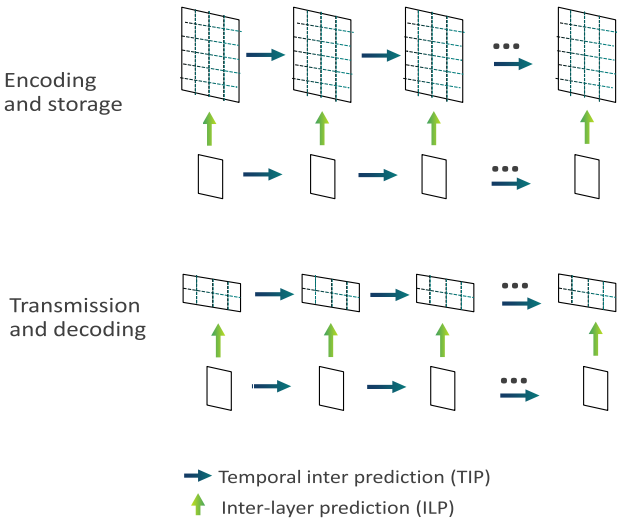

Fig. 17. ***Subpicture-based viewport-dependent 360° video delivery scheme making use of inter-layer prediction.***

a subprofile with a subset of the tools/features contained in an existing VVC profile, by just going through a simple identifier registration process (as specified by Rec. ITU-T T.35 [33]).

VVC version 1 defines six profiles: 1) two single-layer video profiles, the Main 10 profile and the Main 10 4:4:4 profile, which basically support all the coding tools but restrict the bitstream to contain only one layer (although there is no restriction on temporal scalability support of sublayers); 2) two multilayer video profiles, the Multilayer Main 10 profile and the Multilayer Main 10 4:4:4 profile, with the only difference compared with the two single-layer video profiles being that the bitstream can contain multiple layers; and 3) two still picture profiles, the Main 10 Still Picture profile and the Main 10 4:4:4 Still Picture profile, with the only difference compared with the two single-layer video profiles being that the bitstream can contain only one picture, which needs to be intra-picture coded.

V. VVC CODING EFFICIENCY

A. Objective

The JVET has specified some common test conditions (CTCs) [34] to conduct experiments in a well-defined manner to allow for a fair comparison of the outcome of experiments. The CTCs were used to evaluate the proposals during VVC development. The CTC definition includes three mandatory test conditions, reflecting all-intra, random access, and low-delay settings, and the random access case is considered more important than the others due to its much broader usage in applications. A set of 18 video sequences, including Classes A1 and A2 (3840 × 2160), Class B (1920 × 1080), Class C (832 × 480), Class D (416 × 240), Class E (1280 × 720), and Class F (variant resolution), is employed in the experiments. Classes A–D represent

camera-captured video, Class E has video conferencing sequences, and Class F has screen content sequences. Class E is not tested for the random access case (since that case has a higher delay than would be acceptable for video conferencing). Class A sequences are not tested in the low-delay case (since such source material would seldom be used in low-delay applications). All of these test materials are progressively scanned and use 4:2:0 color sampling with 8 or 10 bits per sample. For the random access case, the structural delay is set to 16 frames, and the IRAP random access interval is set to be approximately 1 s. Four rate points are tested with constant QP settings, with the base QP set to 22, 27, 32, and 37 and with the QP of higher temporal sublayers derived using fixed offsets from these values. The experiments in this article use the JVET CTC conditions and the Bjøntegaard delta bit rate (BD-rate) measurement method [35], [36] to evaluate the compression performance based on the following weighted average of peak signal-to-noise ratio (PSNR) values per color component:

$$\mathrm{PSNR}_{\mathrm{YUV}} = \frac{1}{8}(6 * \mathrm{PSNR}_{\mathrm{Y'}} + \mathrm{PSNR}_{\mathrm{C_B}} + \mathrm{PSNR}_{\mathrm{C_R}}).$$

The heavier weighting of $\mathrm{PSNR}_{\mathrm{Y'}}$ is to somewhat compensate for the fact that most of the bits are used to encode the luma component of the video pictures (and it is the most perceptually important component).

In this article, the coding efficiencies of VVC, HEVC, and AVC are compared. A more detailed comparison of HEVC coding efficiency with its predecessors can be found in [19]. In the experiments, the respective reference software encoders were used, that is, the VVC Test Model (VTM-9.0) [37], the HEVC Test Model (HM-16.20) [38], the HEVC SCC Extension Test Model (SCM-8.8) [39] for Class F, and the AVC Joint Test Model (JM-19.0) [40]. Due to level constraints on the DPB capacity for AVC, a random access configuration with a structural delay of eight frames is employed, which can be found in the "cfg/HM-like" configuration files in the JM software package. Average BD-rate savings of VVC over AVC and HEVC for each class of sequences are tabulated in Table 3. The overall average BD-rate savings are based on class A/B/C/E test sequences, which are considered as representing target user scenarios for VVC. It can be seen that the BD-rate savings of VVC over AVC for random access reach 65% on average with up to 72% for 4k resolutions. Compared with HEVC in various configurations, VVC provides the highest coding gain in the random access case, where an average 36.9% YUV BD-rate saving is achieved, and for test sequences with 4k resolutions, the savings is more than 40%. For low-delay and all-intra configurations, VVC achieves 31.1% and 25.5% average YUV BD-rate savings, respectively. For Class F, representing SCC, VVC achieves even higher BD-rate savings when comparing the VTM to the HM in HEVC Main 10 profile configuration. While Main 10 is the most deployed HEVC profile, higher coding efficiency for screen content can be achieved with the HEVC Screen-Extended Main 10 profile, introduced in version 4 of HEVC (see Section III-B4). Comparing the VTM to the SCM in HEVC Screen-Extended Main 10 profile configuration (the last row of the table), VVC still provides 26.4% YUV BD-rate savings for random access, 32.5% YUV BD-rate savings for low delay, and 17.8% YUV BD-rate savings for all-intra. Fig. 18 shows an example of random access rate-distortion plots of VVC, HEVC, and AVC, for the *CatRobot1* UHD video test sequence.

A so-called *tool-off* test has been used to investigate the impact of new coding tools within different modules of VVC. In a tool-off test, a specific set of tools is turned off in VTM-9.0, while all other new coding features remain enabled, and the results are compared with those of VTM-9.0 with all tools turned on. Table 4 shows the gain of the inter-picture coding tools (affine, SBTMVP, AMVR, GPM, BDOF, CIIP, MMVD, BCW, DMVR, and SMVD), intra-picture coding tools (MIP, MRL, ISP, and CCLM), transform & quantization tools (DQ, MTS, LFNST, SBT, and

Table 3 YUV BD-Rate Savings of VVC (VTM-9.0) Over AVC and HEVC

Sequences Class	AVC High 10 Random Access	HEVC Main 10 Random Access	HEVC Main 10 Low Delay B	HEVC Main 10 All Intra
A1 (3840×2160)	71.6%	39.2%	–	30.4%
A2 (3840×2160)	69.8%	42.9%	–	28.6%
B (1920×1080)	65.0%	37.2%	31.8%	23.0%
C (832×480)	55.6%	30.3%	28.1%	22.3%
E (1280×720)	–	–	34.0%	25.7%
Average	**64.8%**	**36.9%**	**31.1%**	**25.5%**
D (416×240)	52.0%	27.5%	24.4%	17.5%
F (screen content)	64.9%	42.4%	43.1%	39.8%
F (screen content) Screen-Extended Main 10	–	26.4%	32.5%	17.8%

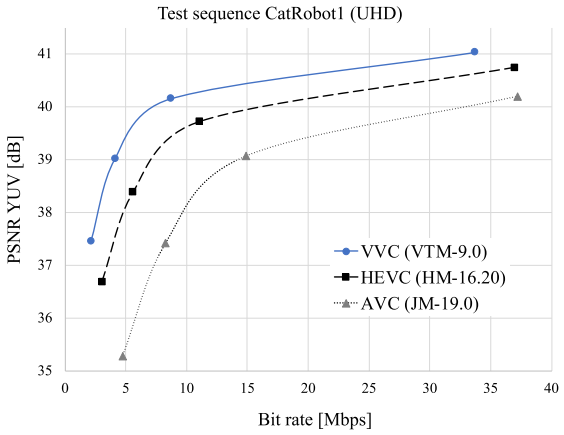

Fig. 18. ***Rate-distortion plots of VVC, HEVC, and AVC for the CatRobot1 video test sequence (random access configuration).***

Table 4 Random Access YUV BD-Rate Savings of VVC (VTM-9.0) Over VVC Without Specific Tool Sets

	Inter	Intra	Trans. & Quant.	Loop Filtering	All 4
A1	10.5%	7.7%	4.8%	5.5%	27.3%
A2	17.1%	2.4%	4.6%	10.1%	31.3%
B	11.5%	2.9%	4.8%	9.7%	26.2%
C	8.9%	3.0%	3.9%	5.5%	20.2%
Avg.	**11.7%**	**3.8%**	**4.5%**	**7.8%**	**25.8%**
EncT	**199%**	**112%**	**127%**	**114%**	**513%**
DecT	**118%**	**98%**	**98%**	**108%**	**142%**
D	9.5%	2.3%	3.8%	5.2%	19.4%
F	7.8%	4.5%	3.9%	4.8%	19.9%

Table 5 MOS and PSNR-YUV BD-Rate Savings of VVC (VTM-10.0) Over HEVC (HM-16.22) and of an Optimized VVC Encoder (VVenC-0.1) Over VTM

UHD SDR	VTM vs. HM		VVenC vs. VTM	
Sequences	MOS	PSNR-YUV	MOS	PSNR-YUV
DrivingPOV3	61%	43%	11%	−12%
Marathon2	37%	33%	7%	−16%
MountainBay2	37%	39%	3%	0%
NeptuneFountain3	38%	27%	21%	−14%
TallBuildings2	41%	37%	16%	−5%
Average	**43%**	**36%**	**12%**	**−9%**

JCCR), and loop filtering tools (ALF, CC-ALF, and LMCS) for the random access case. The coding gain of the VVC QT+MTT block partitioning scheme can be approximated by comparing the first version of the VTM [41], which is basically adding QT+MTT on top of HEVC, to HM-16.20. VTM-1.0 provides around 10% YUV BD-rate savings for random access over the HEVC HM. It should be noted that some VVC coding features, for example, the improved CABAC engine, transform coefficient coding, intra-picture prediction mode coding, and PDPC, cannot be turned off in the VVC reference software. Hence, their respective gains are not included in this experiment. Table 4 further lists relative encoding and decoding runtimes for the averages, where 100% represents the runtime of the respective anchor. The presented results show that VVC's coding efficiency improvement over HEVC stems from multiple new coding features in each major module. In addition, the combined gains of all four tool sets (inter, intra, transform and quantization, and loop filtering) are just slightly lower than the sum of the individual gains. An additional *tool-on* test, where each specific tool set is enabled on top of a version of VTM with all tools off, has been performed as well and the results are not significantly different than for the tool-off test.

B. Subjective

The compression capability goal of the HEVC and VVC projects has been to reduce the bit rate for a given level of subjective video quality, that is, the quality perceived by human observers. While PSNR is a convenient objective measurement method, it is not an adequate substitute for subjective quality measurement. This motivated the JVET to initiate formal testing activities using rigorous subjective assessment methods in order to verify the coding efficiency of the final standard. The first such verification test was completed in October 2020, covering UHD SDR content in a random access configuration, as may be used in newer streaming and broadcast television applications [42]. Here, five challenging UHD SDR sequences outside the JVET test set were selected and encoded over a range of five quality levels spanning from annoying to almost imperceptible impairments. Although the main focus was on comparing the VVC reference software VTM with the HEVC reference software (HM), an open-source encoder implementation (VVenC) was also included in the tests as well [43]. The tested VVenC version 0.1 in "medium" preset runs significantly faster (110×) than VTM and additionally includes subjective quality enhancement techniques, that is, temporal filtering of the input video and perceptually tuned bit allocation [44]. Table 5 summarizes the subjective mean opinion score (MOS) and objective PSNR-YUV-based BD-rate savings for all five test sequences. This test verifies that the VTM and VVenC encoders for VVC significantly improve compression, with the VTM reducing the bit rate by 43% on average relative to the HM for the same perceived quality and VVenC reducing the bit rate by an additional 12% relative to the VTM. On the other hand, the PSNR-YUV BD-rate savings are much lower and even negative (i.e., a bit rate increase) for VVenC versus the VTM. For both tested VVC encoders, the measured subjective quality benefit relative to the HM somewhat exceeds the benefit measured by PSNR-YUV BD-rate numbers—a phenomenon that was also observed for HEVC relative to its AVC predecessor [19]. Fig. 19 shows pooled results for all five test sequences by plotting

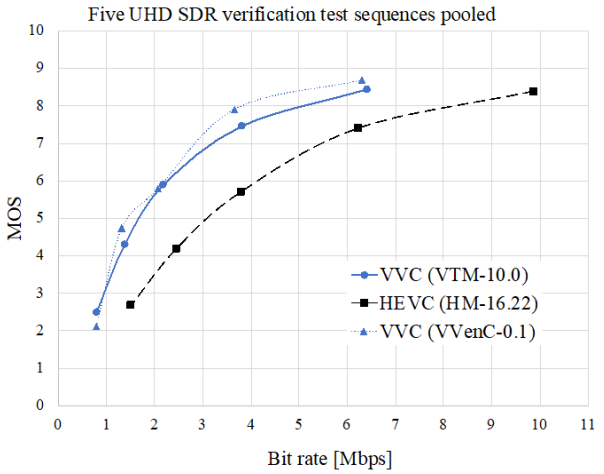

Fig. 19. ***Average (arithmetic) MOS and (geometric mean) bit rates of VVC (VTM and VVenC encoders) and HEVC (HM encoder) pooled over the five UHD SDR sequences used in the verification test.***

the arithmetic average of the MOS values over the geometric average of the corresponding rate points. It can be seen that the quality levels of the VTM and HM are well matched. At the time of writing, testing of HD SDR (random access and low delay), HDR, and 360° video content is ongoing and expected to be completed in April 2021 [45].

VI. CONCLUSION AND OUTLOOK

VVC is a major advance in both video compression capability and the versatility of the application domain, again demonstrating about 50% bit rate reduction for equal subjective quality—a characteristic that it shares with its HEVC and AVC predecessors as a new milestone generation of video coding technology. In terms of applications, it has substantial new features for such uses as the coding of HDR and 360° video content, streaming with adaptive picture resolution, support for compressed-domain bitstream extraction and merging, and, practically, all of the features of the prior international video coding standards and their extensions (e.g., extended chroma formats, scalability, multiview coding, and SCC). Optimized encoder and decoder implementations of VVC have begun to emerge and have clearly demonstrated that the standard is feasible to implement with good compression performance and practical levels of complexity. While the first version of VVC has included only bit depths up to 10 bits per sample, the first extension work for VVC has begun to extend it to support higher bit depths and enhance its performance in the very high (near lossless) fidelity range.

Further research will result in further improvements in video compression, but it may be difficult to significantly surpass the capability of the VVC design for quite a few years to come. Artificial intelligence technologies have shown great promise in that direction, but this work has just begun to emerge, and such techniques are typically difficult to implement at the high speeds and low costs that are necessary for widespread deployment in many video applications. Another promising direction is the development of improved methods of measuring perceptual video quality. Given some improved method of measuring quality, there may be improved compression technologies that can optimize that quality. Yet another interesting direction is the concept of video coding for machines, where the key difference compared with conventional video coding is that the decoded video quality measurement needs to take into account the performance of a nonhuman usage of the decoded video for some particular purposes, for example, by self-driving vehicles.

The breadth of applications of video coding technology also continues to expand, as in recent and emerging work on the coding of point clouds, textures mapped onto moving 3-D meshes, and plenoptic light field coding. Such technologies will bring new requirements to the compression technology although the VVC standard seems quite flexible to address the stable and well-understood applications that have driven the current demand for a new international standard. ■

Acknowledgment

The authors would like to thank the experts of ITU-T VCEG, ISO/IEC MPEG, and their ITU-T/ISO/IEC Joint Video Experts Team (JVET) for their contributions. Their work has not only led to the development of the new Versatile Video Coding (VVC) standard but also made a large archive of innovative contributions available for further study. The archive of JVET documents can be found online at https://www.jvet-experts.org/.

REFERENCES

[1] *High Efficiency Video Coding*, Recommendation ITU-T H.265 and ISO/IEC 23008-2 (HEVC), ITU-T and ISO/IEC JTC 1, Apr. 2013.

[2] *Advanced Video Coding for Generic Audio-Visual Services*, Recommendation ITU-T H.264 and ISO/IEC 14496-10 (AVC), ITU-T and ISO/IEC JTC 1, May 2003.

[3] G. J. Sullivan and T. Wiegand, "Video compression—From concepts to the H.264/AVC standard," *Proc. IEEE*, vol. 93, no. 1, pp. 18–39, Jan. 2005.

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

[5] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 688–703, Jul. 2003.

[6] *Versatile Video Coding*, Recommendation ITU-T H.266 and ISO/IEC 23090-3 (VVC), ITU-T and ISO/IEC JTC 1, Jul. 2020.

[7] *Versatile Supplemental Enhancement Information Messages for Coded Video Bitstreams*, Recommendation ITU-T H.274 and ISO/IEC 23002-7 (VSEI), ITU-T and ISO/IEC JTC 1, Jul. 2020.

[8] Cisco Systems, "Cisco visual networking index: Forecast and trends, 2017–2022," Cisco Syst., White Paper, 2019. [Online]. Available: http://web.archive.org/web/20181213105003/https:/www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf

[9] *Video Codec for Audiovisual Services at P x 64 kbit/s*, Recommendation ITU-T H.261, ITU-T, 1993.

[10] *Codecs for Videoconferencing Using Primary Digital Group Transmission*, Recommendation ITU-T H.120, ITU-T, 1993.

[11] *Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1,5 Mbit/s—Part 2: Video*, ISO/IEC 11172-2, ISO/IEC JTC 1, 1993.

[12] *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video*, Recommendation ITU-T H.262 and ISO/IEC 13818-2, ITU-T and ISO/IEC JTC 1, 1995.

[13] *Video Coding for Low Bit Rate Communication*, Recommendation ITU-T H.263, ITU-T, Mar. 1996.

[14] *Information Technology—Coding of Audio-Visual Objects—Part 2: Visual*, document ISO/IEC 14496-2, ISO/IEC JTC 1, 2001.

[15] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical B pictures and MCTF," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Toronto, ON, Canada, Jul. 2006, pp. 1929–1932.

[16] J. Pfaff *et al.*, "Data-driven intra-prediction modes in the development of the versatile video coding standard," *ITU J. ICT Discoveries*, vol. 3, no. 1, May 2020.

[17] *Information Technology—Digital Compression and Coding of Continuous-Tone Still Images—Part 1: Requirements and Guidelines*, Recommendation ITU-T T.81 and ISO/IEC 10918-1, ITU-T and ISO/IEC JTC 1, 1992.

[18] 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.

[19] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards—Including High Efficiency Video Coding (HEVC)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1669–1684, Dec. 2012.

[20] R. Sjöberg *et al.*, "Overview of HEVC high-level syntax and reference picture management," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1858–1870, Dec. 2012.

[21] *Information Technology—Coding of Audio-Visual Objects—Part 12: ISO Base Media File Format*, document ISO/IEC 14496-12, ISO/IEC JTC 1, 2004.

[22] *Information Technology—Dynamic Adaptive*

Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats, document ISO/IEC 23009-1, ISO/IEC JTC 1, 2012.

[23] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, "An overview of tiles in HEVC," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 969–977, Dec. 2013.

[24] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl, "Tile based HEVC video for head mounted displays," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, San Jose, CA, USA, Dec. 2016, pp. 399–400.

[25] C. Ching Chi *et al.*, "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1827–1838, Dec. 2012.

[26] 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.

[27] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian, "Overview of SHVC: Scalable extensions of the high efficiency video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 20–34, Jan. 2016.

[28] G. Tech, Y. Chen, K. Muller, J.-R. Ohm, A. Vetro, and Y.-K. Wang, "Overview of the multiview and 3D extensions of high efficiency video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 35–49, Jan. 2016.

[29] 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.

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

[31] B. Bross *et al.*, "General video coding technology in responses to the joint call for proposals on video compression with capability beyond HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1226–1240, May 2020.

[32] H. S. Malvar, G. J. Sullivan, and S. Srinivasan, "Lifting-based reversible color transformations for image compression," *Proc. SPIE* vol. 7073, Aug. 2008, Art. no. 707307, Paper 7073-07.

[33] *Procedure for the Allocation of ITU-T Defined Codes for Non-Standard Facilities*, document Recommended ITU-T T.35, ITU-T, 1988.

[34] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, *JVET Common Test Conditions and Software Reference Configurations for SDR Video*, document JVET-N1010, 14th Meeting of ITU-T/ISO/IEC Joint Video Experts Team (JVET), Mar. 2019.

[35] G. Bjøntegaard, *Improvement of BD-PSNR Model*, document VCEG-AI11 of ITU-T SG16/Q6, Berlin, Germany, Jul. 2008. [Online]. Available: http://wftp3.itu.int/av-arch/video-site/0807_Ber/

[36] *Working Practices Using Objective Metrics for Evaluation of Video Coding Efficiency Experiments*, document ITU-T HSTP-VID-WPOM and ISO/IEC DTR 23002-8, ITU-T and ISO/IEC JTC 1, 2020.

[37] *VVC Reference Software Version 8.0*. Accessed: Feb. 2020. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tags/VTM-8.0

[38] *HEVC Reference Software Version 16.20*. Accessed: Sep. 2018. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jct-vc/HM/-/tags/HM-16.20

[39] *HEVC Screen Content Coding Extension Reference Software version 16.21+SCM8.8*. Accessed: Mar. 2020. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jct-vc/HM/-/tags/HM-16.21+SCM-8.8

[40] *AVC Reference Software Version 19.0*. Accessed: Mar. 2019. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jct-vc/JM/-/tags/JM-19.0

[41] J. Chen and E. Alshina, *Algorithm Description for Versatile Video Coding and Test Model 1 (VTM 1)*, document JVET-J1002, 10th Meeting of ITU-T/ISO/IEC Joint Video Experts Team (JVET), Apr. 2018.

[42] V. Baroncini and M. Wien, *VVC Verification Test Report for UHD SDR Video Content*, document JVET-T2020, 21th Meeting of ITU-T/ISO/IEC Joint Video Experts Team (JVET), Oct. 2020.

[43] *Fraunhofer HHI VVenC Software Repository*. Accessed: Sep. 2020. [Online]. Available: https://github.com/fraunhoferhhi/vvenc

[44] A. Wieckowski *et al.*, *Open Optimized VVC Encoder (VVenC) and Decoder (VVdeC) Implementations*, document JVET-T0099, 21th Meeting of ITU-T/ISO/IEC Joint Video Experts Team (JVET), Oct. 2020.

[45] M. Wien, V. Baroncini, A. Segall, and Y. Ye, *VVC Verification Test Plan (Draft 4)*, document JVET-T2009, 21st Meeting of ITU-T/ISO/IEC Joint Video Experts Team (JVET), Oct. 2020.

ABOUT THE AUTHORS

Benjamin Bross (Member, IEEE) received the Dipl.Ing. degree in electrical engineering from RWTH Aachen University, Aachen, Germany, in 2008.

In 2009, he joined the Fraunhofer Institute for Telecommunications–Heinrich Hertz Institute, Berlin, Germany, where he is currently the Head of the Video Coding Systems group and a part-time Lecturer with the HTW Berlin University of Applied Sciences. Since 2010, he has been very actively involved in the ITU-T Video Coding Experts Group (VCEG)|ISO/IEC MPEG video coding standardization processes as a Technical Contributor, a Coordinator of core experiments, and the Chief Editor of the High Efficiency Video Coding (HEVC) standard (ITU-T H.265|ISO/IEC 23008-2) and the Versatile Video Coding (VVC) standard (ITU-T H.266|ISO/IEC 23090-3). Besides giving talks about recent video coding technologies, he is the author or a coauthor of several fundamental HEVC-related publications and the author of two book chapters on HEVC and inter-picture prediction techniques in HEVC.

Mr. Bross received the IEEE Best Paper Award of the 2013 IEEE International Conference on Consumer Electronics–Berlin in 2013, the SMPTE Journal Certificate of Merit in 2014, and the Emmy Award at the 69th Engineering Emmy Awards in 2017 as a part of the Joint Collaborative Team on Video Coding for its development of HEVC.

Jianle Chen (Senior Member, IEEE) received the B.S. and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 2001 and 2006, respectively.

He was formerly with Samsung Electronics, Suwon, South Korea, Qualcomm, San Diego, CA, USA, and Huawei, Santa Clara, CA, USA, focusing on the research of video technologies. Since 2006, he has been actively involved in the development of various video coding standards, including the High Efficiency Video Coding (HEVC) standard, its scalable, format range and screen content coding extensions, and, most recently, the Versatile Video Coding (VVC) standard in the Joint Video Experts Team (JVET). He has been the main developer of the recursive partitioning structure with large block size, which is one of the key features of the HEVC standard and its potential successors. He is currently the Director of the Multimedia R&D Group, Qualcomm, Inc. His research interests include video coding and transmission, point cloud coding, AR/VR, and neural network compression.

Dr. Chen was an Editor of the HEVC specification version 2 (the scalable HEVC (SHVC) text specification) and SHVC Test Model. For VVC, he has been the Lead Editor of the Joint Exploration Test Model (JEM) and the VVC Test Model (VTM). He is an Editor of the VVC Text Specification.

Jens-Rainer Ohm (Member, IEEE) has been holding the chair position of the Institute of Communication Engineering, RWTH Aachen University, Aachen, Germany, since 2000. He is currently the Dean of the Faculty of Electrical Engineering and Information Technology, RWTH Aachen University. Since 1998, he has been participating in the work of the Moving Picture Experts Group (MPEG). He has authored textbooks on multimedia signal processing, analysis, and coding on communication engineering and signal transmission and numerous articles in these fields. His research and teaching activities cover the areas of multimedia signal processing, analysis, compression, transmission, and content description, including 3-D and VR video applications, biosignal processing and communication, application of deep learning approaches in the given fields, and fundamental topics of signal processing and digital communication systems.

Dr. Ohm has been chairing/cochairing various standardization activities in video coding, namely the MPEG Video Subgroup 2002–2018, the Joint Video Team (JVT) of MPEG and ITU-T SG 16 Video Coding Experts Group (VCEG) 2005–2009, and the Joint Collaborative Team on Video Coding (JCT-VC) since 2010 and the Joint Video Experts Team (JVET) since 2015. He has served on the editorial boards of several journals and program committees of various conferences.

Gary J. Sullivan (Fellow, IEEE) received the B.S. and M.Eng. degrees from the University of Louisville, Louisville, KY, USA, 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, Redmond, WA, USA. He has been the Long-standing Chairman/Co-Chairman of various video and image coding standardization activities in ITU-T Video Coding Experts Group (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 the Lead Designer of the DirectX Video Acceleration (DXVA) video decoding feature of the Microsoft Windows operating system.

Dr. Sullivan is 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 Best Paper Awards, and the SMPTE Digital Processing Medal. The team efforts that he has led have been recognized by three Emmy Awards.

Ye-Kui Wang received the B.S. degree in industrial automation from Beijing Institute of Technology, Beijing, China, in 1995, and the Ph.D. degree in information and telecommunication engineering from the Graduate School in Beijing, University of Science and Technology of China, Hefei, China, in 2001.

His earlier working experiences and titles include the Chief Scientist of Media Coding and Systems, Huawei Technologies, San Diego, CA, USA, the Director of Technical Standards with Qualcomm, San Diego, a Principal Member of Research Staff with Nokia Corporation, Tampere, Finland, and so on. He is currently a Principal Scientist with Bytedance Inc., San Diego. He has been an active contributor to various multimedia standards, including video codecs, file formats, RTP payload formats, and multimedia streaming and application systems, developed by various standardization organizations, including ITU-T Video Coding Experts Group (VCEG), ISO/IEC Moving Picture Experts Group (MPEG), Joint Video Team (JVT), JVT on Video Coding (JCT-VC), JCT-3V, 3GPP SA4, IETF, AVS, DVB, ATSC, and DECE. He has coauthored about 1000 standardization contributions and over 50 academic articles. He is a listed inventor for more than 300 U.S. patents. His research interests include video coding, storage, transport, and multimedia systems.

He has been chairing the development of OMAF at MPEG and an editor for several standards, including Versatile Video Coding (VVC), Versatile Supplemental Enhancement Information (VSEI), OMAF, all versions of High Efficiency Video Coding (HEVC), VVC file format, HEVC file format, layered HEVC file format, ITU-T H.271, SVC file format, MVC, RFC 6184, RFC 6190, RFC 7798, 3GPP TR 26.906, and 3GPP TR 26.948.