Overview of the Emerging HEVC Screen Content Coding Extension

Jizheng Xu, *Senior Member, IEEE*, Rajan Joshi, *Member, IEEE*, and Robert A. Cohen, *Senior Member, IEEE*

***Abstract*— A screen content coding (SCC) extension to High Efficiency Video Coding (HEVC) is currently under development by the Joint Collaborative Team on Video Coding, which is a joint effort from the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. The main goal of the HEVC-SCC standardization effort is to enable significantly improved compression performance for videos containing a substantial amount of still or moving rendered graphics, text, and animation rather than, or in addition to, camera-captured content. This paper provides an overview of the technical features and characteristics of the current HEVC-SCC test model and related coding tools, including intra-block copy, palette mode, adaptive color transform, and adaptive motion vector resolution. The performance of the SCC extension is compared against existing standards in terms of bitrate savings at equal distortion.**

***Index Terms*— High Efficiency Video Coding (HEVC), screen content coding (SCC), video coding.**

I. Introduction

IN JANUARY 2013, the first edition of the High Efficiency Video Coding (HEVC) standard, also known as HEVC version 1 [1], was finalized. This work was done by the Joint Collaborative Team on Video Coding (JCT-VC), which was jointly established in 2010 by ITU-T Study Group 16 [Video Coding Experts Group (VCEG)] and ISO/IEC JTC1/SC 29/WG 11 [Moving Picture Experts Group (MPEG)]. One of the primary requirements of that standard was that it demonstrates a substantial bitrate reduction over the existing H.264/AVC [2] standard. Both the initial development of HEVC and the earlier development of H.264/AVC focused on compressing camera-captured video sequences. Although several different test sequences were used during the development of these standards, the camera-captured sequences exhibited common characteristics such as the presence of sensor noise and an abundance of translational motion. Recently, however, there has been a proliferation of applications that use video devices to display more than just camera-captured content. These applications include displays that combine camera-captured and computer graphics, wireless displays, tablets, automotive displays, screen sharing, and so on [3]. The type of video content used in these applications can contain a significant amount of stationary or moving computer graphics and text, along with camera-captured content. However, unlike camera-captured content, screen content frequently contains no sensor noise, and such content may have large uniformly flat areas, repeated patterns, highly saturated or a limited number of different colors, and numerically identical blocks or regions among a sequence of pictures. These characteristics, if properly leveraged, can offer opportunities for significant improvements in compression efficiency over a coding system primarily designed for camera-captured content.

During the development of HEVC, decisions as to which tools to incorporate into the version 1 standard were primarily made based on their coding performance on camera-captured content. Several tool proposals showed that the characteristics of screen content video can be leveraged to obtain additional improvements in compression efficiency over the HEVC version 1 standard under development.

Residual scalar quantization (RSQ) and base colors and index map (BCIM) [4] were proposed early during the HEVC development process. Because screen content often has high contrast and sharp edges, RSQ directly quantized the intra-prediction residual, without applying a transform. BCIM took advantage of the observation that the number of unique colors in screen content pictures is usually limited compared with camera-captured content. RSQ and BCIM could, respectively, be considered early forms of transform skip, which is part of HEVC version 1, and palette mode, which is described in Section III-B. Additional modes such as transform bypass where both the transform and quantization steps are bypassed for lossless coding and the use of differential pulse code modulation (DPCM) for sample-based intra prediction were proposed in [5].

Because screen content often contains repeated patterns, dictionary and Lempel–Ziv coding tools were shown to be effective at improving coding efficiency, especially on pictures containing text and line graphics [6]–[8]. These tools store a dictionary of previously coded samples. If a block or coding unit (CU) contains strings of samples that are already contained in the dictionary, then a pointer to the dictionary can be signaled, thus avoiding the need to transform and quantize a prediction residual.

In January 2014, the MPEG Requirements subgroup published a set of requirements for an extension of HEVC for coding of screen content [3]. This document identified three types of screen content: 1) mixed content; 2) text and graphics with motion; and 3) animation. Up to lossless coding

Manuscript received January 15, 2015; revised June 20, 2015; accepted September 10, 2015. Date of publication September 14, 2015; date of current version January 6, 2015. This paper was recommended by Associate Editor T. Wiegand.

J. Xu is with Microsoft Research, Beijing 100080, China (e-mail: jzxu@microsoft.com).

R. Joshi is with Qualcomm Technologies, Inc., Andover, MA 01810 USA.

R. A. Cohen is with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2478706

performance was specified, for RGB and YUV 4:4:4 video having 8 or 10 bits per color component. Given the earlier published drafts of these requirements, along with the existing evidence that additional tools could improve the coding performance of HEVC on screen content material, VCEG and MPEG issued a joint call for proposals (CfP) for coding of screen content [9].

At the next JCT-VC meeting in March 2014, seven responses to the CfP were evaluated. After identifying the tools that produced significant improvements in performance for screen content, several core experiments (CEs) were defined. The tools included in these CEs were intra-block copying extensions, line-based intra copy, palette mode, string matching for sample coding, and cross-component prediction (CCP) and adaptive color transforms (ACTs). After evaluating the outcome of the CEs and related proposals, the HEVC Screen Content Coding (SCC) Draft Text 1 [10] was published in July 2014.

This paper gives an overview of the tools and coding performance associated with this new HEVC-SCC extension. At the time of this writing, the current version of the text is Draft 2 [11]. Section II describes SCC support in HEVC version 1 and HEVC range extensions (HEVC-RExt). New coding tools in the HEVC-SCC extension are described in Section III. Section IV presents new encoding algorithms introduced during the development of HEVC-SCC. Coding performance comparisons are discussed in Section V, and Section VI concludes this paper.

II. Screen Content Coding Support in HEVC

The HEVC-SCC extension is developed based on HEVC version 1 [1] and HEVC-RExt [12]. Thus, it inherits the coding structure and coding tools of HEVC version 1 and HEVC-RExt. HEVC-SCC also maintains backward compatibility to HEVC version 1 and HEVC-RExt. As an example, an HEVC-SCC decoder can decode HEVC version 1 bitstreams and the decoded videos are identical to those produced by an HEVC version 1 decoder. Screen content, as one class of video, was also considered during the development of HEVC version 1 and HEVC-RExt, although it was not the main focus of those standards. In the following sections, the structures of HEVC and HEVC-RExt, along with the coding tools that primarily targeted screen content, are briefly introduced.

A. HEVC Version 1

HEVC version 1 follows the conventional hybrid coding structure as in previous video coding standards. An input image is first divided into image blocks, referred to as coding tree units (CTUs), of predefined size. The size of a CTU can be of 16×16, 32×32, or 64×64 luma samples. A CTU contains corresponding chroma samples according to the input color format. A quad-tree split, with a CTU as the root, divides the CTU into one or more CUs. A CU is square shaped and can have 8×8, 16×16, 32×32, or 64×64 luma samples and corresponding luma samples. A CU can be classified as an intra CU or an inter CU, and it is further divided into one, two, or four prediction units (PUs). For an intra CU, spatially neighboring reconstructed samples are used to predict its PUs. For an inter CU, a motion vector is sent for each PU, which is used by a motion compensation process to generate the prediction from other pictures.

A transform quad tree is also defined for a CU to indicate how the predicted residuals are decorrelated with spatial transforms after the prediction process. A leaf of a transform quad tree is referred to as a transform unit (TU). An integer transform is applied to each TU. All transforms applied to an inter CU and most transforms applied to an intra CU are derived to approximate the discrete cosine transform of appropriate size. The only exception is the 4×4 transform applied to an intra CU, which is designed to approximate the discrete sine transform.

During the development of HEVC version 1, it was observed that for screen content, spatial transform mentioned above do not always help in improving coding efficiency. For most screen content, the images are sharp, containing irregular edges and shapes. After the prediction process, the residual signal may already be sparse because the background can be precisely predicted while irregular foreground may not. In such a case, the existing transform will spread the energy to most frequencies instead of compacting the energy, thereby destroying the sparsity of the residual, leading to low coding efficiency during entropy coding. Thus, for those blocks, skipping the transform and quantizing data in the spatial domain can be a better choice, as was demonstrated for H.264/AVC in [13]. HEVC version 1 can skip the transform for a 4×4 TU, whether it is intra [14] or inter [15]. This transform skip is equivalent to applying an identity transform to the TU. Thus, the quantization process after applying transform skip is the same as that applied after the spatial transform. It turns out that such a simple design can lead to a significant coding efficiency improvement for screen content, e.g., the bit saving brought by the transform skip mode is about 7.5% for typical 4:2:0 screen content [15]. When applied to 4:4:4 screen content, the coding gain for transform skip is much larger [16], ranging from 5.5% to 34.8%.

B. HEVC-RExt

After HEVC version 1, HEVC-RExt was developed to support non-4:2:0 color formats, e.g., 4:4:4 and 4:2:2v, and high-bit-depth video, e.g., up to 16 bits. Because most screen content is captured in the 4:4:4 color format, which is not supported by HEVC version 1, more attention was given to coding of screen content in HEVC-RExt. The coding tools that improved the coding efficiency for screen content in HEVC-RExt compared with HEVC version 1 include the following.

1) Improvements to Transform Skip Mode: As mentioned above, HEVC version 1 supports transform skip only for 4×4 TUs. HEVC-RExt extends transform skip to all TUs, regardless of their size [17]. Enabling transform skip for all TUs has two benefits. One is that the coding efficiency for screen content can be further improved. The other is that encoders have the flexibility to exploit the transform skip mode. For example, a specific encoder may support only

large TUs so that the encoding complexity can be reduced. If transform skip is allowed only for 4×4 TUs, the performance of such an encoder would be adversely affected since it cannot exploit the benefit brought by transform skip, which can be much more notable for screen content. Other improvements include coefficient coding for transform skip blocks. For an intra transform block, when a spatial transform is applied to the residual, the energy is compacted at the top-left corner of the block. However, when transform skip is used on a transform block, because its prediction is from the adjacent top row and/or left column of samples, the prediction error is higher at the bottom-right corner compared with the top-left corner. The different characteristics of the residual of transform skip blocks relative to transformed blocks make entropy coding designed for transformed blocks inefficient when applied to transform skip blocks. A simple yet effective solution adopted in HEVC-RExt is to rotate the residual of transform skip blocks by 180° [18], [19]. This rotation process is applied only to 4×4 intra transform skip blocks [20]. Another improvement for the entropy coding of the transform skip blocks is that a single context is used to code the significant coefficient map. For transformed blocks, the context used for significance coding depends on the position (frequency) of the coefficient. However, since all the coefficients of a transform skip block are from the residual in the spatial domain, using a single context is more reasonable than using different contexts depending on the position.

2) Residual Differential Pulse Code Modulation: Even after intra prediction, there is still correlation in the residual signal, which can be exploited. Residual DPCM (RDPCM) predicts the current residual using its immediately neighboring residual. In HEVC-RExt, RDPCM was proposed for intra lossless coding [21]. Then, it was extended to lossy coding [22] and inter coding [23]. In RDPCM, the left reconstructed residual or the above one is used to predict the current residual, depending on whether the block uses horizontal or vertical RDPCM. RDPCM is used only for transform skip blocks. In inter mode, flags are sent to indicate whether RDPCM is used and if so, its direction. In contrast, RDPCM in intra mode is applied in the same direction as the intra-prediction direction. Because of this, using the reconstructed residual to predict the current residual is identical to using reconstructed samples to predict the current sample in intra mode (if clipping of the reconstructed sample is ignored). From this perspective, RDPCM shortens the prediction distance, leading to better prediction. For screen content, a short-distance prediction is quite useful because usually the content is sharp and changes rapidly.

3) Cross-Component Prediction: CCP [24] and its predecessor LM chroma mode [25] were proposed to exploit correlation among color components [24]. In CCP, the residual of the second or third color component can be predicted from the residual of the first color component multiplied by a scaling factor. The factor is selected from $\{0, \pm(1/8), \pm(1/4), \pm(1/2), \pm 1\}$ and is signaled to the decoder. One advantage of performing prediction on residuals in the spatial domain is that the reconstruction process for the second and third color components does not depend on reconstructed samples of the first color component. Hence, the reconstruction process for different color components can be performed in parallel once the residuals for the three color components are available. Although such a design can still leave a certain degree of correlation among color components, it was demonstrated that CCP can significantly improve the coding efficiency when coding videos have the RGB color format. Because much screen content is captured in the RGB domain, CCP is very effective in coding of screen content.

4) Other Improvements: Some other aspects of HEVC-RExt, although not specifically designed for SCC, also improve the coding efficiency for screen content. For example, the initialization of Rice parameters based on the previous similar blocks was primarily designed for high-bit-depth coding, but it also showed improvement for coding screen content [26].

C. HEVC-SCC

Unlike HEVC version 1 and HEVC-RExt, the tools added for the HEVC-SCC extension focus primarily on coding screen content. To illustrate the framework of SCC, an SCC encoder is shown Fig. 1. As shown in Fig. 1, HEVC-SCC is based on the HEVC framework where several new modules/tools are added. The new coding tools are as follows.

1) *Intra-Block Copy (IBC):* HEVC-SCC introduces a new CU mode in addition to the conventional intra and inter modes, referred to as IBC. When a CU is coded in IBC mode, the PUs of this CU find similar reconstructed blocks within the same picture. IBC can be considered as motion compensation within the current picture.
2) *Palette Mode:* For screen content, it is observed that for many blocks, a limited number of different color values may exist. Thus, palette mode enumerates those color values, and then for each sample, sends an index to indicate to which color it belongs. Palette mode can be more efficient than the prediction-then-transform representation.
3) *ACT:* Because much screen content uses the RGB color space, removing inter-color component redundancy is important for efficient coding. Earlier work with adaptive color space transforms was shown to yield improvements in coding efficiency [27]. In HEVC-SCC, a CU-level adaptation is used to convert residual to different color spaces. More precisely, an image block in the RGB color space can be coded directly, or it can be adaptively converted to the YCoCg color space during coding.
4) *Adaptive Motion Vector Resolution (AMVR):* Unlike camera-captured content, where motion is continuous, screen content often has discrete motion, which has a granularity of one or more samples. Thus, for much screen content, it is not necessary to use fractional motion compensation. In HEVC-SCC, a slice-level control is enabled to switch the motion vectors between full-pel and fractional resolutions.

The details of these new tools are described in the following sections.

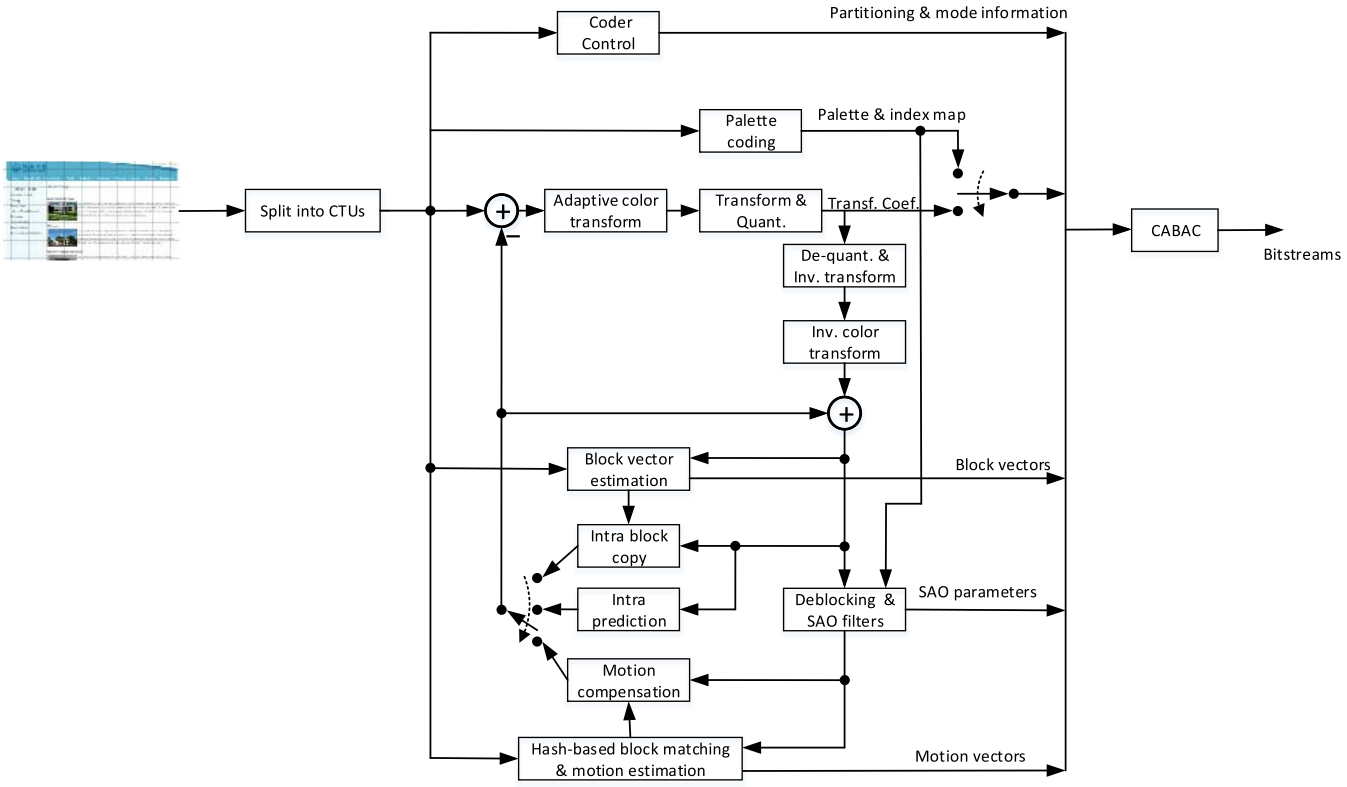


Fig. 1. Encoder for SCC extension.

III. New Coding Tools in HEVC-SCC

This section describes the new coding tools that were adopted into the HEVC-SCC text specifications, i.e., normative specifications. When relevant, some nonnormative aspects are included in this section, and a more complete description of nonnormative aspects of the tools is included in Section IV.

A. Intra-Block Copy

IBC is a coding tool similar to inter-picture prediction. The main difference is that in IBC, a predictor block is formed from the reconstructed samples (before the application of in-loop filtering) of the current picture. Previously, IBC was proposed in the context of AVC/H.264 [28], but the coding gain was not consistently high across different test sequences, which at the time were primarily camera-captured sequences and not screen content material. A CU-based IBC mode was proposed during the HEVC-REext development [29]. A modified version [30] was adopted into the HEVC-REext text specification, but was later removed. IBC has been a part of HEVC-SCC test model since the beginning of the HEVC-SCC development.

At the early stage of HEVC-SCC development, IBC was performed at the CU level. A block vector was coded to specify the location of the predictor block. However, since both IBC and inter mode share the concept of vectors representing displaced blocks, it is natural to unify the design of IBC and inter mode. Methods to unify these modes [31], [32] have shown that using the inter-mode syntax design for IBC is also an adequate choice.

In the current HEVC-SCC design, IBC is performed at the PU level and is treated as an inter PU. Specifically, using the inter-mode design, the current picture can also be used as a reference picture for IBC. When a PU's reference picture is the current picture, it means that its prediction is performed from the reconstructed samples, before in-loop filtering in the encoder, of the current picture, which corresponds to the original IBC design. When the current picture is used as a reference, it is marked as a long-term reference. After the current picture is fully decoded, the reconstructed picture after in-loop filtering is added to the decoded picture buffer (DPB) as a short-term reference, which is identical to what HEVC version 1 does after decoding a picture. Using the inter-mode design to enable IBC at the PU level enables greater flexibility in combining IBC and inter mode. For example, an inter CU can have two PUs, one using conventional inter mode and the other using IBC; a PU can be bidirectionally predicted from an average between a block from the current picture and a block from a previous picture.

However, unification between IBC and inter mode does not mean that IBC can be directly implemented as an inter mode in practice. The implementation of IBC can be much different from that of inter mode in many platforms. Such differences exist because when the current picture is used as a reference, it has not been fully reconstructed, whereas the other reference pictures have been decoded and stored in the DPB.

For IBC mode, because the block to be processed and its prediction are from the same picture, several constraints have been placed to avoid other modules from getting adversely affected. The constraints for IBC mode are as follows.

1) The predictor block may not overlap the current CU, which avoids generating predictions from unreconstructed samples.

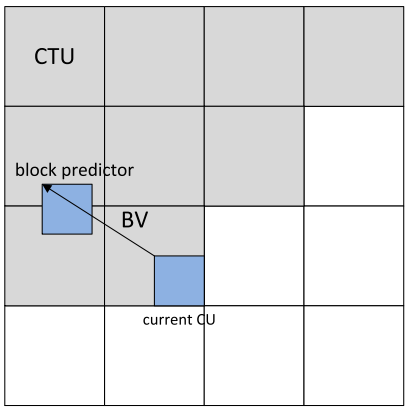


Fig. 2. Search area for IBC (shown in gray).

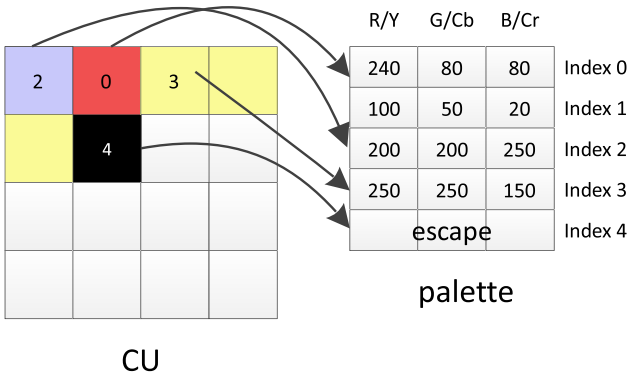


Fig. 3. Palette example (palette size = 4).

2) The predictor block and the current CU should be within the same slice and the same tile. Otherwise, there will be dependency among different slices or tiles, which affects the parallel processing capability provided by the design of slices and tiles.
3) The predictor block is normatively required to be entirely contained in the search region shown in Fig. 2. It is designed so as to avoid affecting the parallel processing capability provided by wavefronts. To simply the design, the constraint still holds even when wavefronts are not being used.
4) The block vector precision is full-pel.

B. Palette Mode

Palettes are an efficient method for representing blocks containing a small number of distinct color values. Rather than apply a prediction and transform to code a block, palette mode signals indices to indicate the color values of each sample. An early use of palettes was in the conversion of 24-bit RGB images to 8-bit index images to save on RAM or video memory buffer space. A CU-based palette coding mode was proposed during the HEVC-RExt development [33]. The palette mode was further refined through CEs and *ad hoc* group discussions and was adopted [34] into the SCC text specification.

A palette refers to a table consisting of representative color values from the CU coded using the palette mode. For each sample in the CU, an index into the current table is signaled in the bitstream. The decoder uses the palette table and the index to reconstruct each sample of the CU. Each entry in the palette table consists of three components (RGB or YCbCr). For 4:2:0 and 4:2:2 color formats, if no chroma components are present for the current sample position, only the first component is used for reconstruction. A special index, known as an escape index, is reserved to indicate that a sample does not belong to the palette. In such a case, in addition to coding the escape index, the quantized values of the component(s) of the escape sample are also coded in the bitstream.

The size of the palette table is referred to as the palette size. If the palette size is nonzero, then the indices from zero to palette size minus one are reserved for denoting the entries from the palette, and the escape index is set to be equal to the palette size. An example for this coding is shown in Fig. 3. On the encoder side, palette table derivation is performed. This is further discussed in Section IV-C. Normative aspects of palette coding can be divided into two categories: 1) coding of the palette table and 2) coding of the palette indices of the samples in the CU.

The palette needs to be signaled to the decoder. It is typical that a palette shares some entries with neighboring blocks that are coded using the palette mode. To exploit this sharing, the palette table is coded using a combination of prediction from the palette entries of the previously coded CUs and new palette entries that are explicitly signaled. For prediction, a predictor palette consisting of palette entries from the previous CUs coded in palette mode is maintained. For each entry in the palette predictor, a flag is signaled to specify whether that entry is reused in the current palette. The collection of flags is signaled using run-length coding of zeros. The run lengths are coded using exponential Golomb code of order 0. After signaling the predicted palette entries, the number of new palette entries and their values are signaled. This is shown in Fig. 4. In this example, the predictor palette contains six entries. Out of these, three are reused in the current palette (zeroth, second, and third). These are assigned indices 0, 1, and 2, respectively, in the current palette. This is followed by two new entries.

For the first CTU of each slice, tile, and CTU row (when wavefronts are used), the palette predictor is initialized using initialization entries signaled in the picture parameter set or to zero. After a CU has been coded in palette mode, the palette predictor is updated as follows. First, all the entries from the current palette are included in the predictor palette. This is followed by all the entries that were not reused from the previous predictor palette. This process is continued till all the entries from the previous predictor palette that were not reused are included or the maximum palette predictor size is reached. The updated predictor palette is illustrated on the right side of Fig. 4.

To code the palette indices, first, a flag, palette_escape_val_present_flag, is coded to indicate whether there are any escape indices present in the current CU. Two different scans may be used to code the palette indices in a CU, namely, horizontal traverse scan and vertical traverse scan, as shown

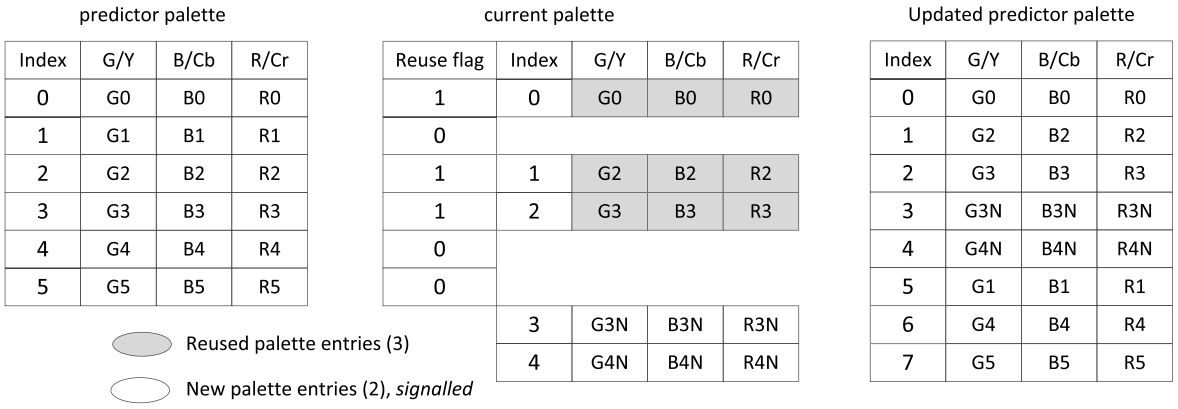


Fig. 4. Construction of palette from the predictor palette and new explicitly signaled entries.

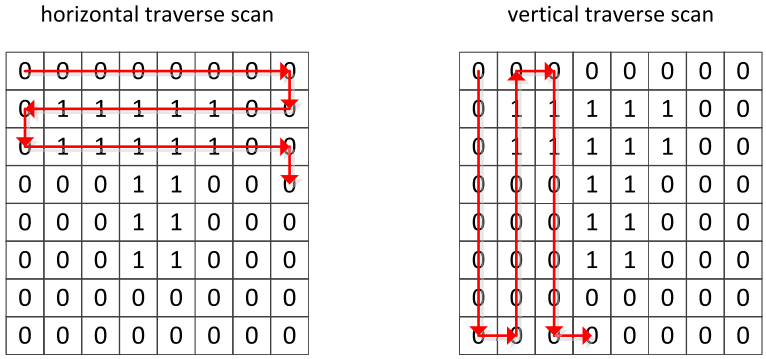


Fig. 5. Horizontal and vertical traverse scans for the coding of palette indices.

in Fig. 5. In the following description, a horizontal traverse scan is assumed. If the scan is vertical traverse, the CU index map may be transposed before coding (or after decoding). The direction of the scan is signaled using palette_transpose_flag.

Since screen content may typically contain flat areas with uniform or near-uniform sample values, run-length coding of palette indices is an efficient method of compression. In addition, it is observed that indices in consecutive rows or columns may be identical. To exploit this, each palette sample may be coded in one of the two palette sample modes, COPY_INDEX_MODE and COPY_ABOVE_MODE, which are signaled in the bitstream as palette_run_type_flag. In COPY_INDEX_MODE, the palette index is coded followed by a run value that specifies the number of subsequent samples that have the same index. The index values are coded using a truncated binary code [35]. In COPY_ABOVE_MODE, the palette index is copied from the previous row. This is followed by a run value that specifies the number of subsequent positions for which the index is copied from the previous row. Both COPY_INDEX_MODE and COPY_ABOVE_MODE runs may span multiple rows. In COPY_ABOVE_MODE, only the run value is signaled but not the index. If the index for a particular sample corresponds to the escape index, the component value(s) are quantized and coded. Fig. 6 shows an example of palette sample modes, indices, and run values for a 4×4 block.[1]

[1]The 4×4 block is chosen for convenience. The minimum palette block size is 8×8.

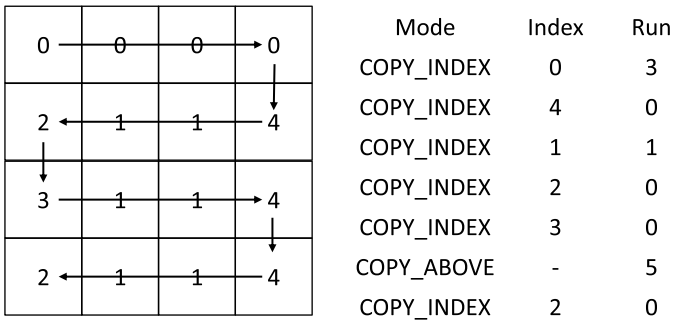


Fig. 6. Example of palette sample modes, indices, and run values for a 4×4 block.

The run coding uses a concatenation of unary code and exponential Golomb code of order zero. The code can be described as follows. A run of zero is represented as 0. A run of length $L \geq 1$ is represented as a concatenation of 1 and the exponential Golomb code (order zero) representation for $(L - 1)$. Both prefix and suffix are truncated based on the maximum possible run value when the run continues to the end of the block. The code is specified in Table I.

Several redundancies are exploited in the coding of palette indices to reduce the number of syntax elements and make the coding more efficient. For example, when the palette size is equal to 0 or when the palette size is equal to 1 and there are no escape indices present, then the index values can be inferred, thereby eliminating the

TABLE I
BINARIZATION FOR THE PALETTE RUN VALUE

value	*prefix*	*suffix*
0	0	-
1	10	-
2-3	110	X
4-7	1110	XX
7-15	11110	XXX
...	...	...

need to signal the palette sample modes, indices, and runs. Furthermore, a COPY_ABOVE_MODE may not occur in the first row (or column for vertical traverse scan) and may not follow COPY_ABOVE_MODE. This is used in inferring COPY_INDEX_MODE under these conditions.

C. Adaptive Color Transform

Much screen content is captured in the RGB color space. For an image block in the RGB color space, usually, there can be strong correlation among different color components such that a color space conversion is useful for removing inter-color component redundancy. However, for screen content, there may exist many image blocks containing different features having very saturated colors, which leads to less correlation among color components. For those blocks, coding directly in the RGB color space may be more effective. To handle different characteristics of image blocks in screen content, an RGB-to-YC_oC_g conversion [36] as shown in the following equation was investigated, and it turned out to be effective:

$$\begin{bmatrix} Y \\ C_o \\ C_g \end{bmatrix} = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 0 & -1/2 \\ -1/4 & 1/2 & -1/4 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{1}$$

When this color transform is used, both the input image block and its corresponding prediction use the same conversion. Because the conversion is linear, it is identical to having the transform applied to residuals in the spatial domain when the prediction processes in different color components are consistent. Thus, in HEVC-SCC, the conversion is applied on the residual [37], which makes the prediction process for different color components independent. It is also noted that for intra-coded blocks, when the intra-prediction directions for different color components are not the same, the color transform is not allowed to be used. This limitation is specified because when the intra-prediction directions are different, the correlation among colocated samples across color components is decreased, making the color transform less effective. The color transform also changes the norm of different components. To normalize the errors in different color spaces, when the above transform is used for an image block, a set of QP offsets $(-5, -5, -3)$ is applied to those three color components [38] during quantization. After the quantization and reconstruction, an inverse transform is applied to the quantized residual so that the reconstruction is still kept in the input color space.

To limit the dynamic range expansion brought by the color transform, a lifting-based approximation is used in HEVC-SCC. The corresponding color space is called YC_oC_g–R. The forward and inverse transforms from RGB to YC_oC_g–R, also from [36], are

$$\begin{aligned} C_o &= R - B \\ t &= B + \lfloor C_o/2 \rfloor \\ C_g &= G - t \\ Y &= t + \lfloor C_g/2 \rfloor \end{aligned} \tag{2}$$

$$\begin{aligned} t &= Y - \lfloor C_g/2 \rfloor \\ G &= C_g + t \\ B &= t - \lfloor C_o/2 \rfloor \\ R &= B + C_o \end{aligned} \tag{3}$$

where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x.

With the forward transform (2), the bit depth of Y remains the same as that of the input, while for C_o and C_g components, the bit depth will be increased by one. For lossless coding, (2) and (3) are directly used. While for lossy coding, to keep the bit depth identical to the residual of the original color space, an additional right shift is applied to the C_o and C_g components after the forward transform. Correspondingly, an additional left shift is applied before the inverse transform [39].

Even after the ACT is applied in the encoder, or in cases when the input video has already undergone a color transform, there can still be inter-component redundancy that may benefit from the application of CCP, e.g., as shown in [40] and [41]. Therefore, CCP may also be applied to CUs that have undergone the ACT. It is shown in [42] that both ACT and CCP can significantly improve the coding efficiency for videos in the RGB color space, while ACT might improve more. Lai *et al.* [43] compare performance with different adaptive strategies, which indicates that coding in the YCoCg color space remarkably improves the coding efficiency, while adaptively choosing a better color space can further improve the performance, especially for RGB videos and lossless coding cases.

D. Adaptive Motion Vector Resolution

For camera-captured video, the movement of a real-world object is not necessarily exactly aligned to the sample positions in a camera's sensor. Motion compensation is therefore not limited to using integer sample positions, i.e., fractional motion compensation is used to improve compression efficiency. Computer-generated screen content video, however, is often generated with the knowledge of the sample positions, resulting in motion that is discrete or precisely aligned with sample positions in the picture. For this kind of video, integer motion vectors may be sufficient for representing the motion. Savings in bitrate can be achieved by not signaling the fractional portion of the motion vectors.

In HEVC-SCC, AMVR [44] defines a slice-level flag to indicate that the current slice uses integer (full-pel) motion vectors for luma samples. If the flag is true, then the motion vector predictions, motion vector differences, and resulting motion vectors assume only integer values, so the bits representing the fractional values do not need to be signaled.

TABLE II
TYPES OF CANDIDATE BVS AND ENCODER RESTRICTIONS ON THEIR USE

candidate BVs	PU size restriction	other restrictions
local search region	–	current CTU and one or more CTUs to the left (configurable)
extended search region	up to 16×16	enable or disable (configurable)
Previously used BVs with same PU size	up to 16×16	up to 64 BVs

To minimize the differences between HEVC-SCC and HEVC version 1, the decoded integer motion vectors are simply left shifted by one or two bits, so the rest of the motion vector processing is unchanged as the fractional bits are zero.

IV. ENCODING ALGORITHMS

Because of the introduction of new coding tools as described in Section III and different characteristics of screen content, new encoding algorithms were introduced during the development of HEVC-SCC. These methods are part of HEVC-SCC test model 3 [45]. It should be understood that these methods are not normative, but help improve the coding efficiency of the HEVC-SCC coding tools. The new algorithms that are much different from conventional video encoding methods are described in this session for a better understanding of HEVC-SCC.

A. Intra-Block Vector Search

In order to decide whether to use IBC mode for a CU, a rate-distortion (RD) cost is calculated for the CU. Block matching (BM) is performed at the encoder to find the optimal BV for each PU. Depending on the size of the PU, the following three types of candidate BVs are evaluated as shown in Table II.

The RD cost for each candidate BV is evaluated as

$$\text{RD\_cost}_Y = \text{SAD}_{\text{luma}} + \lambda \times \text{BV}_{\text{bits}} \tag{4}$$

where SAD is the sum of absolute differences between the samples in a block and corresponding samples in the matching block, and BV_{bits} is the number of bits needed to signal the BV. Note that only the SAD for the luma block is used at this stage. Once the RD cost for all the candidate vectors is evaluated, the four candidate BVs with the least RD cost are chosen. The candidate with the best RD cost is chosen as the optimal BV. In this case, the RD cost is evaluated based on both luma and chroma as follows:

$$\text{RD\_cost}_{\text{YUV}} = \text{SAD}_{\text{luma}} + \text{SAD}_{\text{chroma}} + \lambda \times \text{BV}_{\text{bits}}. \tag{5}$$

For 8×8 PUs, the entire picture region conforming to the restrictions on the predictor block as described in Section III-A is searched for the optimal BV using a hash-based search. Each node in the hash table records the position of each BV candidate in the picture. Only the block vector candidates that have the same hash entry value as that of the current block are examined. For PU sizes up to 16×16, with the exception of 8×8, an extended area may be searched for the optimal BV. In this case, block vectors in the region conforming to the restrictions on the predictor block as described in Section III-A with at least one zero component (horizontal or vertical) are considered as candidate BVs.

The 16-bit hash entries for the current block and the reference block are calculated using the original sample values. Let grad_{BLK} denote the gradient of the 8×8 block, and let dc0, dc1, dc2, and dc3 denote the dc values of the four 4×4 sub-blocks of the 8×8 block. Then, the 16-bit hash entry H is calculated as

$$\begin{aligned} H = {} & \text{msb}(\text{dc0}, 3) \ll 13 + \text{msb}(\text{dc1}, 3) \ll 10 \\ & + \text{msb}(\text{dc2}, 3) \ll 7 + \text{msb}(\text{dc3}, 3) \ll 4 \\ & + \text{msb}(\text{grad}_{\text{BLK}}, 4) \end{aligned} \tag{6}$$

where $\text{msb}(X, n)$ represents the n most significant bits of X.

The gradient is calculated as follows. First, for each sample in the block except for the first line and first column, grad$X$ is calculated as the absolute difference between the current sample and the sample to the left. Similarly, grad$Y$ is calculated as the absolute difference between the current sample and the sample above. Then, grad for the sample is set equal to the average of grad$X$ and grad$Y$. The final grad_{BLK} for the block is the cumulative sum of the per-sample grad values over the block.

B. Inter Search

Large motion is often observed in screen content, which makes the conventional motion search computationally prohibitive. However, because digitally captured screen content is usually noiseless, it is possible to use fast search methods in the encoder to identify the corresponding block in a reference picture. In the reference software, a hash-based BM algorithm [46] is used to search for identical blocks in reference pictures. For a reference picture, a 16-bit cyclic redundancy check (CRC) value is calculated using the original sample values for a block starting from any sample position and having sizes of 8×8, 16×16, 32×32, and 64×64. An 18-bit hash value is formed comprising the 16-bit CRC and 2 bits representing the block size. An inverted index is used to index every block having the same hash value. For a CU with a $2N \times 2N$ PU, the hash value is calculated, and with this hash value, an exact match can be searched for using the inverted index in linear time, regardless of the search range and the number of reference pictures. To reduce the memory requirement for storing the hash table, a prefiltering process is applied to exclude blocks that can be easily predicted using intra prediction. For those blocks, it is not necessary to perform BM. When a block cannot find an exact match in the reference picture, as is the case for most camera-captured material, conventional motion estimation is used. For simple screen content material, most blocks can have exact matches. Thus, the motion estimation search process is often be skipped, which can significantly reduce the encoding complexity.

TABLE III
TEST SEQUENCES (4:4:4) IN THE CTC

Resolution	Sequence name	Category
1920x1080	sc_flyingGraphics_1920x1080_60_8bit	TGM
	sc_desktop_1920x1080_60_8bit	TGM
	sc_console_1920x1080_60_8bit	TGM
	MissionControlClip3_1920x1080_60p_8b444	M
	EBURainFruits_1920x1080_50_10bit	CC
	Kimono1_1920x1080_24_10bit	CC
1280x720	sc_web_browsing_1280x720_30_8bit	TGM
	sc_map_1280x720_60_8bit	TGM
	sc_programming_1280x720_60_8bit	TGM
	sc_SlideShow_1280x720_20_8bit	TGM
	sc_robot_1280x720_30_8bit	A
2560x1440	Basketball_Screen_2560x1440_60p_8b444	M
	MissionControlClip2_2560x1440_60p_8444	M

TABLE IV
TEST SEQUENCES (4:2:0) IN THE CTC

Resolution	Sequence name	Category
1920x1080	sc_flyingGraphics_1920x1080_60_8bit_420	TGM
	sc_desktop_1920x1080_60_8bit_420	TGM
	sc_console_1920x1080_60_8bit_420	TGM
	MissionControlClip3_1920x1080_60p_8b420	M
1280x720	sc_web_browsing_1280x720_30_8bit_420_r1	TGM
	sc_map_1280x720_60_8bit_420	TGM
	sc_programming_1280x720_60_8bit_420	TGM
	sc_SlideShow_1280x720_20_8bit_420	TGM
	sc_robot_1280x720_30_8bit_420	A
2560x1440	Basketball_Screen_2560x1440_60p_8b420	M
	MissionControlClip2_2560x1440_60p_8420	M
1024x768	ChinaSpeed_1024x768_30	A

C. Palette Mode Encoding

To evaluate the RD cost of encoding a CU in palette mode, a palette table for the CU is derived as follows. A modified k-means clustering method is used to derive a palette table in the lossy case. The palette table is initialized to have no entries. Then, for each sample in the CU, the nearest palette entry (in terms of SAD) is determined. If the SAD is within a threshold value, the sample is added to the cluster corresponding to the nearest palette entry. Otherwise, a new palette entry is created to be equal to the sample value. After processing all the samples in the CU, each palette entry is updated by the centroid of the cluster. Then, the palette entries are sorted based on the number of entries in their associated clusters. There is a limit on the maximum palette size, which is signaled in the sequence parameter set. If the number of palette entries exceeds this limit, the least frequent clusters are eliminated and the sample values belonging to those clusters are converted to escape samples.

As discussed in Section III-B, if a palette entry is already in the palette predictor, the cost of including it in the current palette is small. On the other hand, when the centroid is not in the palette predictor, all the components have to be explicitly signaled in the bitstream, resulting in a much higher cost. Hence, an RD analysis is performed to determine whether assigning a cluster to an existing palette predictor entry has a lower RD cost. If this is the case, then the centroid is replaced by the palette predictor entry. After this step, duplicate palette entries are removed. Also, if a cluster includes a single entry that is not contained in the palette predictor, the entry is

TABLE V
COMPARISON OF SCM-3.0 WITH VERSUS WITHOUT IBC (BD-RATE CHANGE)

		AI	RA	LB
RGB	TGM	-31.3%	-19.1%	-10.6%
	M	-28.9%	-19.2%	-8.9%
	A	-1.1%	-0.5%	-0.1%
	CC	-0.1%	0.0%	0.0%
YUV	TGM	-32.5%	-19.1%	-9.8%
	M	-29.5%	-19.9%	-8.9%
	A	-1.6%	-0.5%	0.1%
	CC	-0.2%	0.0%	0.1%

TABLE VI
COMPARISON OF SCM-3.0 WITH VERSUS WITHOUT PALETTE (BD-RATE CHANGE)

		AI	RA	LB
RGB	TGM	-15.5%	-10.5%	-6.8%
	M	-3.7%	-2.6%	-1.5%
	A	0.0%	-0.1%	0.0%
	CC	0.0%	0.0%	0.0%
YUV	TGM	-16.2%	-11.1%	-6.8%
	M	-5.9%	-4.4%	-2.7%
	A	0.1%	0.2%	0.1%
	CC	0.0%	0.1%	0.1%

TABLE VII
COMPARISON OF SCM-3.0 WITH VERSUS WITHOUT ACT (BD-RATE CHANGE)

		AI	RA	LB
RGB	TGM	-9.6%	-11.6%	-11.1%
	M	-16.6%	-23.1%	-23.7%
	A	-24.5%	-24.9%	-24.0%
	CC	-24.5%	-27.5%	-26.1%
YUV	TGM	-0.4%	-0.7%	-1.0%
	M	0.1%	0.4%	0.4%
	A	0.1%	-0.1%	0.0%
	CC	0.1%	0.5%	0.3%

TABLE VIII
COMPARISON OF SCM-3.0 WITH VERSUS WITHOUT AMVR (BD-RATE CHANGE)

		RA	LB
RGB	TGM	-1.4%	-2.2%
	M	0.0%	0.0%
	A	0.0%	0.0%
	CC	0.0%	0.0%
YUV	TGM	-1.5%	-2.4%
	M	0.0%	-0.1%
	A	0.0%	0.0%
	CC	0.0%	0.0%

TABLE IX
LOSSY CODING PERFORMANCE COMPARISON BETWEEN SCM-4.0 AND HM-16.4 FOR 4:4:4 SEQUENCES (BD-RATE CHANGE)

Colour format	Category	AI			RA			LB		
		G/Y	B/U	R/V	G/Y	B/U	R/V	G/Y	B/U	R/V
RGB	TGM	-64.5%	-60.9%	-62.0%	-56.9%	-51.0%	-53.1%	-50.8%	-42.9%	-45.3%
	M	-54.8%	-49.7%	-49.5%	-50.2%	-42.4%	-42.0%	-41.7%	-29.2%	-28.2%
	A	-26.3%	-19.5%	-16.8%	-26.2%	-17.3%	-12.9%	-24.4%	-11.9%	-5.5%
	CC	-25.6%	-5.5%	-10.4%	-28.3%	-5.8%	-14.4%	-26.1%	-1.6%	-11.9%
YUV	TGM	-57.4%	-61.3%	-62.8%	-48.0%	-52.6%	-55.3%	-40.5%	-44.9%	-47.4%
	M	-45.2%	-50.9%	-50.8%	-36.7%	-45.1%	-44.8%	-23.8%	-33.6%	-33.3%
	A	-1.2%	-10.9%	-7.6%	-0.4%	-10.1%	-6.8%	0.0%	-7.0%	-4.9%
	CC	0.4%	0.0%	-0.2%	0.6%	-0.2%	-0.3%	0.6%	-0.3%	-0.2%

TABLE X
LOSSY CODING PERFORMANCE COMPARISON BETWEEN SCM-4.0 AND HM-16.4 FOR 4:2:0 SEQUENCES (BD-RATE CHANGE)

Category	AI			RA			LB		
	G/Y	B/U	R/V	G/Y	B/U	R/V	G/Y	B/U	R/V
TGM	-49.0%	-49.3%	-50.5%	-39.4%	-40.6%	-42.2%	-32.7%	-33.2%	-34.4%
M	-36.6%	-37.6%	-37.6%	-29.4%	-31.2%	-31.5%	-18.0%	-18.7%	-19.5%
A	-7.3%	-11.7%	-10.7%	-3.8%	-12.4%	-9.9%	-2.0%	-8.2%	-5.7%

converted to an escape sample. For lossless coding, a slightly different method of palette derivation based on the histogram of the CU samples is used.

Once the palette table and assignment of CU sample values to palette indices are determined, a rate-based encoder optimization is performed to determine the palette_run_type_flag and run values [47]. For example, assuming that index mode is chosen for the sample, a run value is determined. Then, the per-sample cost in bits for coding the index and run value is calculated. This cost is compared with the per-sample bit cost for coding the run value assuming that COPY_ABOVE_MODE is chosen. The run type with the lower per-sample bit cost is chosen. The decision is greedy in the sense that sometimes when COPY_ABOVE_MODE is chosen, the cost of coding an index is just postponed to a future run. More sophisticated strategies are possible. For example, instead of first performing index assignment and then making the decision about palette_run_type_flag and run values, it may be possible to make these decisions jointly. As a simple example, by forcing a sample value to map to a different index, it may be possible to extend a run, thereby lowering the RD cost. Such strategies are not currently considered in HEVC-SCC test model 3 encoder.

TABLE XI
LOSSLESS CODING PERFORMANCE COMPARISON BETWEEN SCM-4.0 AND HM-16.4 FOR 4:4:4 SEQUENCES (BD-RATE CHANGE)

Colour format	Category	AI	RA	LB
RGB	TGM	-45.8%	-35.2%	-32.2%
	M	-24.3%	-6.3%	-3.9%
	A	-4.4%	-1.1%	-1.1%
	CC	-0.2%	0.4%	0.4%
YUV	TGM	-46.7%	-36.4%	-33.3%
	M	-23.9%	-6.3%	-3.8%
	A	-1.7%	-0.3%	-0.3%
	CC	0.0%	0.0%	0.0%

TABLE XII
LOSSLESS CODING PERFORMANCE COMPARISON BETWEEN SCM-4.0 AND HM-16.4 FOR 4:2:0 SEQUENCES (BD-RATE CHANGE)

Category	AI	RA	LB
TGM	-34.1%	-23.9%	-21.1%
M	-21.6%	-6.0%	-3.5%
A	-0.7%	-0.2%	-0.1%

D. Decision for Adaptive Color Transform

When the ACT is enabled, a CU can choose to perform the color transform described in Section III-C on the residual of three color components. In the current test model, the encoder simply compares the RD costs of coding in both modes (with or without the color transform) and selects one with the least cost. Thus, the encoding time increases accordingly.

E. Decision for Adaptive Motion Vector Resolution

For each slice, an encoder can choose to use full-pel or fractional motion vectors for luma samples. In the current test model, instead of checking both options and comparing the RD costs, a fast algorithm is used to make the decision for the entire slice. The basic idea of this algorithm is to determine whether most blocks in the current slice have exact matching blocks with full-pel displacements in the reference picture.

Similar to the inter-search scheme described earlier, hash values are precomputed for every possible 8×8 block in the reference picture. Next, the current slice is divided into nonoverlapped 8×8 blocks. For each block, the corresponding hash value is calculated, and an exact match for each current block's hash is searched for in the list of reference block hashes, without considering the case of hash collision. A match indicates that an integer motion vector would yield an exact

TABLE XIII
LOSSY CODING PERFORMANCE COMPARISON BETWEEN SCM-4.0 AND JM-18.6 FOR 4:4:4 SEQUENCES (BD-RATE CHANGE)

Colour format	Category	AI			RA			LB		
		G/Y	B/U	R/V	G/Y	B/U	R/V	G/Y	B/U	R/V
RGB	TGM	-86.1%	-83.5%	-84.1%	-80.4%	-76.1%	-77.4%	-77.7%	-73.0%	-74.4%
	M	-80.1%	-76.2%	-76.0%	-74.3%	-68.2%	-67.3%	-69.7%	-60.8%	-59.8%
	A	-52.4%	-45.0%	-40.1%	-54.8%	-49.5%	-43.2%	-56.4%	-51.0%	-43.1%
	CC	-58.4%	-35.6%	-44.3%	-63.3%	-42.6%	-51.5%	-60.1%	-36.5%	-48.2%
YUV	TGM	-74.6%	-75.0%	-77.0%	-68.1%	-70.4%	-73.3%	-65.4%	-67.6%	-70.5%
	M	-63.6%	-64.8%	-64.8%	-56.9%	-63.2%	-63.1%	-51.5%	-60.5%	-60.6%
	A	-23.4%	-35.5%	-29.3%	-32.2%	-48.0%	-41.8%	-39.0%	-59.6%	-54.6%
	CC	-26.5%	-18.5%	-25.4%	-40.0%	-42.2%	-42.6%	-39.8%	-51.5%	-53.6%

TABLE XIV
LOSSY CODING PERFORMANCE COMPARISON BETWEEN SCM-4.0 AND JM-18.6 FOR 4:2:0 SEQUENCES (BD-RATE CHANGE)

Category	AI			RA			LB		
	G/Y	B/U	R/V	G/Y	B/U	R/V	G/Y	B/U	R/V
TGM	-69.9%	-64.4%	-65.3%	-62.3%	-61.6%	-62.6%	-60.3%	-58.7%	-59.5%
M	-57.4%	-52.2%	-53.1%	-51.8%	-51.7%	-52.7%	-47.0%	-46.0%	-47.7%
A	-31.7%	-33.3%	-33.5%	-36.4%	-45.2%	-44.2%	-39.5%	-45.0%	-43.9%

match, without the need to actually perform the motion search. If exact matches are found for most of the 8×8 blocks, then full-pel motion vectors will be used for the slice. For the detailed algorithms, readers can refer to [44].

V. PERFORMANCE ANALYSIS

Simulations were conducted to evaluate the new coding tools in HEVC-SCC and compare the coding efficiency of HEVC-SCC with HEVC-REext and H.264/AVC. For H.264/AVC, HEVC-REext, and HEVC-SCC, the reference software JM-18.6, HM-16.4, and SCM-3.0/SCM-4.0 were used, respectively.

A. Test Conditions

The common test conditions (CTCs) used during the development of HEVC-SCC [48] were used to generate the results presented in this section. These test conditions include video sequences comprising a variety of resolutions and content types. Table III lists the 4:4:4 sequences. In the category column, 'TGM, CC, M, and A stand for text and graphics with motion, camera-captured content, mixed content (i.e., content having both TGM and CC), and animation. For each sequence, two color formats are tested. One is in RGB, and the other is in YCbCr (denoted by YUV in the following tables). Table IV lists the 4:2:0 sequences. All of them are in YCbCr color format. As indicated by their names, many of the 4:2:0 sequences are generated from their 4:4:4 correspondences.

Three test configurations, i.e., all intra (AI), random access (RA), and low-delay B (LB), were used. For lossy coding, four QPs {22, 27, 32, 37} were applied. More details can be found in [48].

B. Performance Analysis of Coding Tools

To verify the effectiveness of coding tools described in Section III, Tables V–VIII show the coding performance compared with an anchor not having the specific coding tool in SCM-3.0, for 4:4:4 sequences.[2] Only the BD-rate measures of the first color component are listed due to space limitations. From these Tables V–VIII, it can be observed that both IBC and palette modes are very effective for videos in the TGM and M categories. For camera-captured content, both tools neither help nor harm the coding efficiency notably. Another observation is that the coding gain for AI is larger than that for RA or LB, because both IBC and palette modes are for intra coding.

From Table VII, it can be observed that ACT improves the coding of RGB videos significantly, regardless of categories. In contrast, the performance gains of ACT are limited when the video is already in the YUV color format.

TABLE XV
LOSSLESS CODING PERFORMANCE COMPARISON BETWEEN SCM-4.0 AND JM-18.6 FOR 4:4:4 SEQUENCES (BD-RATE CHANGE)

Colour format	Category	AI	RA	LB
RGB	TGM	-66.7%	-59.1%	-58.9%
	M	-44.4%	-28.4%	-26.6%
	A	-20.9%	-14.3%	-13.0%
	CC	-6.9%	-2.9%	-2.8%
YUV	TGM	-53.5%	-47.7%	-47.4%
	M	-29.3%	-16.7%	-14.9%
	A	-5.0%	-7.6%	-6.2%
	CC	-0.7%	-1.4%	-1.4%

C. Performance Comparison With Existing Standards

Tables XI to XII show the performance comparison between SCM-4.0 and HM-16.4 for lossy and lossless coding, respectively. For lossy coding, the BD-rate changes for all

[2]For individual tool comparisons, SCM-3.0 is used because it already contains all coding tools mentioned above. The improvements of later versions of SCM are mainly for 4:2:0 sequences.

TABLE XVI
LOSSLESS CODING PERFORMANCE COMPARISON BETWEEN SCM-4.0 AND JM-18.6 FOR 4:2:0 SEQUENCES (BD-RATE CHANGE)

Category	AI	RA	LB
TGM	-44.2%	-39.8%	-36.6%
M	-26.9%	-16.9%	-14.1%
A	-4.5%	-9.6%	-7.7%

three color components are shown. For lossless coding, the average bit-saving percentages are listed. Significant improvements in compression efficiency are achieved, especially for videos in the TGM and M categories. Decreases in BD-rate of more than 50% indicate that SCM-4.0 can perform with more than twice the compression efficiency of HM-16.4. For comparisons with H.264/AVC, Tables XIII to XVI show the coding performance comparison between SCM-4.0 and JM-18.6.

VI. CONCLUSION

The HEVC-SCC extension was developed to leverage the unique characteristics of computer-generated and digitally captured videos. The IBC mode takes advantage of the presence of exact copies of blocks between different pictures in a video sequence. Palette mode targets blocks having a limited number of unique color values, which frequently occur in computer-generated pictures. The ACT performs an in-loop color space conversion from RGB to YCoCg, and an AMVR eliminates the need to signal fractional motion vectors for computer-generated videos. With these new tools, HEVC-SCC is capable of performing with more than twice the compression efficiency of HEVC-RExt for materials primarily having text and graphics or mixed content.

ACKNOWLEDGMENT

The authors would like to thank many experts of the aforementioned standardization organizations whose joint efforts and contributions led to the creation of the screen content coding extension.

REFERENCES

[1] G. J. Sullivan, J. 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.

[2] *Advanced Video Coding for Generic Audiovisual Services*, ITU-T and ISO/IEC JTC1, document ITU-T Rec. H.264 and ISO/IEC 14496-10, May 2003.

[3] H. Yu, K. McCann, R. Cohen, and P. Amon, *Requirements for an Extension of HEVC for Coding of Screen Content*, ISO/IEC JTC 1/SC 29/WG 11, document MPEG2014/N14174, San Jose, CA, USA, Jan. 2014.

[4] C. Lan, J. Xu, F. Wu, and G. J. Sullivan, *Screen Content Coding*, document JCTVC-B084, Geneva, Switzerland, Jul. 2010.

[5] W. Gao, G. Cook, M. Yang, J. Song, and H. Yu, *Near Lossless Coding for Screen Content*, document JCTVC-F564, Turin, Italy, Jul. 2011.

[6] C. Lan, X. Peng, J. Xu, and F. Wu, *Intra and Inter Coding Tools for Screen Contents*, document JCTVC-E145, Geneva, Switzerland, Mar. 2011.

[7] S. Wang and T. Lin, *4:4:4 Screen Content Coding Using Macroblock-Adaptive Mixed Chroma-Sampling-Rate*, document JCTVC-H0073, San Jose, CA, USA, Feb. 2012.

[8] T. Lin, P. Zhang, S. Wang, and K. Zhou, *4:4:4 Screen Content Coding Using Dual-Coder Mixed Chroma-Sampling-Rate (DMC) Techniques*, document JCTVC-I0272, Geneva, Switzerland, Apr. 2012.

[9] *Joint Call for Proposals for Coding of Screen Content*, ITU-T Q6/16 Visual Coding and ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio, document MPEG2014/N14175, San Jose, CA, USA, Jan. 2014.

[10] R. Joshi and J. Xu, *HEVC Screen Content Coding Draft Text 1*, document JCTVC-R1005, Sapporo, Japan, Jul. 2014.

[11] R. Joshi and J. Xu, *HEVC Screen Content Coding Draft Text 2*, document JCTVC-S1005, Strasbourg, France, Nov. 2014.

[12] J. Boyce *et al.*, *Edition 2 Draft Text of High Efficiency Video Coding (HEVC), Including Format Range (RExt), Scalability (SHVC), and Multi-View (MV-HEVC) Extensions*, document JCTVC-R1013, Sapporo, Japan, Jul. 2014.

[13] M. Narroschke, "Extending H.264/AVC by an adaptive coding of the prediction error," in *Proc. 25th Picture Coding Symp. (PCS)*, O5-3. Beijing, China, Apr. 2006.

[14] C. Lan, J. Xu, G. J. Sullivan, and F. Wu, *Intra Transform Skipping*, document JCTVC-I0408, Geneva, Switzerland, Apr. 2012.

[15] X. Peng, C. Lan, J. Xu, and G. J. Sullivan, *Inter Transform Skipping*, document JCTVC-I0408, Stockholm, Sweden, Jul. 2012.

[16] R. Cohen and A. Vetro, *AHG8: Impact of Transform Skip on New Screen Content Material*, document JCTVC-L0428, Geneva, Switzerland, Jan. 2013.

[17] X. Peng, J. Xu, L. Guo, J. Sole, and M. Karczewicz, *Non-RCE2: Transform Skip on Large TUs*, document JCTVC-N0288, Vienna, Austria, Jul. 2013.

[18] J. An, L. Zhao, Y.-W. Huang, and S. Lei, *Residue Scan for Intra Transform Skip Mode*, document JCTVC-J0053, Stockholm, Sweden, Jun. 2012.

[19] D. He, J. Wang, and G. Martin-Cocher, *Rotation of Residual Block for Transform Skipping*, document JCTVC-J0093, Stockholm, Sweden, Jun. 2012.

[20] X. Peng, B. Li, and J. Xu, *On Residual Rotation for Inter and Intra BC Modes*, document JCTVC-O0186, Geneva, Switzerland, Oct. 2013.

[21] S. Lee, I.-K. Kim, and C. Kim, *AHG7: Residual DPCM for HEVC Lossless Coding*, document JCTVC-L0117, Geneva, Switzerland, Jan. 2013.

[22] R. Joshi, J. Sole, and M. Karczewicz, *AHG8: Residual DPCM for Visually Lossless Coding*, document JCTVC-M0351, Incheon, Korea, Apr. 2013.

[23] M. Naccari, M. Mrak, A. Gabriellini, S. Blasi, and E. Izquierdo, *Inter-Prediction Residual DPCM*, Incheon, Korea, document JCTVC-M0442, Apr. 2013.

[24] T. Nguyen, A. Khairat, and D. Marpe, *Non-RCE1/Non-RCE2/AHG5/AHG8: Adaptive Inter-Plane Prediction for RGB Content*, document JCTVC-M0230, Incheon, Korea, Apr. 2013.

[25] X. Zhang, C. Gisquet, E. Francois, F. Zou, and O. C. Au, "Chroma intra prediction based on inter-channel correlation for HEVC," *IEEE Trans. Image Process.*, vol. 23, no. 1, pp. 274–286, Jan. 2014.

[26] M. Karczewicz *et al.*, *RCE2: Results of Test D1 on Rice Parameter Initialization*, document JCTVC-O0239, Geneva, Switzerland, Oct. 2013.

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

[28] S.-L. Yu and C. Chrysafis, *New Intra Prediction Using Intra-Macroblock Motion Compensation*, document JVT-C151R1, Fairfax, VA, USA, May 2002.

[29] D.-K. Kwon and M. Budagavi, *AHG8: Video Coding Using Intra Motion Compensation*, document JCTVC-M0350, Incheon, Korea, Apr. 2013.

[30] C. Pang, J. Sole, L. Guo, M. Karczewicz, and R. Joshi, *Non-RCE3: Intra Motion Compensation With 2-D MVs*, document JCTVC-N0256, Vienna, Austria, Jul. 2013.

[31] B. Li and J. Xu, *Non-SCCE1: Unification of Intra BC and Inter Modes*, document JCTVC-R0100, Sapporo, Japan, Jul. 2014.

[32] C. Pang *et al.*, *CE2 Test1: Intra Block Copy and Inter Signalling Unification*, document JCTVC-T0094, Geneva, Switzerland, Feb. 2015.

[33] L. Guo, J. Sole, and M. Karczewicz, *Palette Mode for Screen Content Coding*, document JCTVC-M0323, Incheon, Korea, Apr. 2013.

[34] P. Onno, X. Xiu, Y.-W. Huang, and R. Joshi, *Suggested Combined Software and Text for Run-Based Palette Mode*, document JCTVC-M0348, Sapporo, Japan, Jul. 2014.

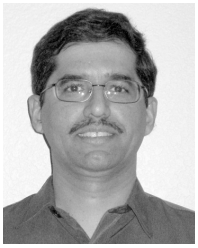
[35] (2014). *Truncated Binary Encoding*. [Online]. Available: http://en.wikipedia.org/wiki/Truncated_binary_encoding

[36] H. S. Malvar, G. J. Sullivan, and S. Srinivasan, "Lifting-based reversible color transformations for image compression," *Optical Engineering+ Applications (pp. 707307-707307)*. International Society for Optics and Photonics, 2008.
[37] L. Zhang *et al.*, *SCCE5 Test 3.2.1: In-Loop Color-Space Transform*, document JCTVC-R0147, Sapporo, Japan, Jul. 2014.
[38] B. Li and J. Xu, *Fix for Adaptive Color Space Coding in JCTVC-Q0035*, document JCTVC-Q0213, Valencia, Spain, Mar. 2014.
[39] L. Zhang, J. Chen, M. Karczewicz, B. Li, and J. Xu, *Unification of Colour Transforms in ACT*, document JCTVC-S0254, Strasbourg, France, Oct. 2014.
[40] A. Khairat, T. Nguyen, M. Siekmann, D. Marpe, and T. Wiegand, "Adaptive cross-component prediction for 4:4:4 High Efficiency Video Coding," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Paris, France, Oct. 2014, pp. 3734–3738.
[41] T. Nguyan, J. Sole, and J. Kim, *RCE1: Summary Report of HEVC Range Extensions Core Experiment 1 on Inter-Component Decorrelation Methods*, document JCTVC-N0034, Vienna, Austria, Jul. 2013.
[42] L. Zhang, J. Chen, J. Sole, M. Karczewicz, X. Xiu, and J.-Z. Xu, "Adaptive color-space transform for HEVC screen content coding," in *Proc. Data Compress. Conf. (DCC)*, Snowbird, UT, USA, Apr. 2015, pp. 233–242.
[43] P. Lai, S. Liu, and S. Lei, *AHG6: On Adaptive Color Transform (ACT) in SCM 2.0*, document JCTVC-S0100, Strasbourg, France, Nov. 2014.
[44] B. Li, J. Xu, G. J. Sullivan, Y. Zhou, and B. Lin, *Adaptive Motion Vector Resolution for Screen Content*, document JCTVC-S0085, Strasbourg, France, Oct. 2014.
[45] R. Joshi *et al.*, *Screen Content Coding Test Model 3 Encoder Description (SCM 3)*, document JCTVC-S1014, Strasbourg, France, Oct. 2014.
[46] B. Li, J. Xu, F. Wu, X. Guo, and G. J. Sullivan, *Description of Screen Content Coding Technology Proposal by Microsoft*, document JCTVC-Q0035, Valencia, Spain, Mar. 2014.
[47] W. Pu, F. Zou, V. Seregin, R. Joshi, M. Karczewicz, and J. Sole, *Non-CE6: Palette Parsing Dependency and Palette Encoder Improvement*, document JCTVC-S0156, Strasbourg, France, Oct. 2014.
[48] H. Yu, R. Cohen, K. Rapaka, and J. Xu, *Common Test Conditions for Screen Content Coding*, document JCTVC-T1015, Geneva, Switzerland, Feb. 2015.

Jizheng Xu (M'07–SM'10) received the B.S. and M.S. degrees in computer science from University of Science and Technology of China, Hefei, China, and the Ph.D. degree in electrical engineering from Shanghai Jiaotong University, Shanghai, China.

He joined Microsoft Research, Beijing, China, in 2003, where he is currently a Lead Researcher. He has been an active contributor to ISO/MPEG and ITU-T video coding standards. He has over 30 technical proposals adopted by H.264/AVC, H.264/AVC scalable extension, High Efficiency Video Coding (HEVC), HEVC range extension, and HEVC screen content coding standards. He has authored or co-authored over 100 conference and journal refereed papers. He holds over 30 U.S. granted or pending patents in image and video coding. His research interests include image and video representation, media compression, and communication.

Dr. Xu chaired and co-chaired the ad-hoc group of exploration on wavelet video coding in MPEG, and various technical ad-hoc groups in JCT-VC, e.g., on screen content coding, on parsing robustness, on lossless coding. He co-organized and co-chaired special sessions on scalable video coding, directional transform, and high quality video coding at various conferences. He also served as the Special Session Co-Chair of the IEEE International Conference on Multimedia and Expo in 2014.



Rajan Joshi (M'95) received the B.Tech. degree in electrical engineering and the M.Tech. degree in communications engineering from IIT Bombay, Mumbai, India, in 1988 and 1990, respectively, and the Ph.D. degree in electrical engineering from Washington State University, Pullman, WA, USA, in 1996.

He was with the Xerox Palo Alto Research Center, Palo Alto, CA, USA, in 1995. From 1996 to 2006, he was a Senior Research Scientist with Eastman Kodak Company, Rochester, NY, USA. From 2006 to 2008, he was a Senior Member of the Technical Staff with Thomson Corporate Research, Burbank, CA, USA. He joined Qualcomm Technologies, Inc., Andover, MA, USA, in 2008, where he is currently a Senior Staff Engineer/ Manager. He has been an active participant in several past and present video and image coding standardization efforts, such the Joint Collaborative Team on Video Coding for the development of the High Efficiency Video Coding standard and its extensions, JPEG2000, and the advanced display stream compression task group in VESA. His current research interests include video and image coding, video processing, and information theory.

Robert A. Cohen (S'85–M'90–SM'12) received the B.S. (*summa cum laude*) and M.Eng. degrees in computer and systems engineering, and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA.

He held positions with IBM, Armonk, NY, USA, and Harris RF Communications, Rochester, NY, USA. From 1991 to 2001, he was a Senior Member of the Research Staff with Philips Research, Briarcliff Manor, NY, USA, where he performed research in areas related to the Grand Alliance HDTV decoder, rapid prototyping for very large scale integration video systems, statistical multiplexing for digital video encoders, scalable MPEG-4 video streaming, and next-generation video surveillance systems. He has been a Principal Member of the Research Staff with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, since 2007, where he performs research, publishes, and generates patents related to next-generation video coding, screen-content coding, and perceptual image/video coding and processing. He also actively participates in ISO/MPEG and ITU standardization activities, including chairing several ad hoc groups and core experiments, contributing to High Efficiency Video Coding-related call for proposals and drafting the Joint Call for Proposals for coding of screen content in JCT-VC. His research interests include video coding and communications, video, image, and signal processing, and 3D point cloud compression.

Dr. Cohen organized the Special Session on Screen Content Coding in PCS 2013. He was a Guest Editor of *Signal Processing: Image Communication* of the Special Issue on Advances in High Dynamic Range Video Research.