|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  20th Meeting: Geneva, CH, 10–18 Feb. 2015 | Document: JCTVC-T0120 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Palette coding mode for non-444 screen content video** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Xiaoyu Xiu, Yan Ye, Yuwen He 9710 Scranton Rd, #250 San Diego, CA 92121, USA | Tel: Email: | + 1-858-210-4830 [Xiaoyu.Xiu@InterDigital.com](mailto:Xiaoyu.Xiu@InterDigital.com)  [Yan.Ye@InterDigital.com](mailto:Yan.Ye@InterDigital.com)  Yuwen.He@InterDigital.com |
| *Source:* | InterDigital Communications, LLC | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This contribution proposes to enable palette coding mode for screen content video with non-4:4:4 chroma format by slightly modifying the current palette coding mode for 4:4:4 format. The proposed method reuses most of the existing palette coding logics for 4:4:4 video. For lossy coding, the proposed method reportedly provides average {Y, Cb, Cr} BD-rate savings of {3.2%, 6.3%, 5.5%}, {2.3%, 6.3%, 5.2%} and {1.5%, 5.1%, 4.1%} for AI, RA and LB , respectively; for lossless coding, the corresponding bit-rate savings of the second method for AI, RA and LB are 0.2%, 0.1% and 0% respectively.

# Introduction

The palette coding design in the latest HEVC screen content coding (SCC) specification draft 2 [1] only supports the input video in 4:4:4 chroma format. Correspondingly, the palette coding mode is always disabled for non-444 input video. However, many other applications with non-444 input may also benefit from the new tools being developed in HEVC SCC extension [2]. Therefore, it is highly desirable to support the palette coding mode for non-4:4:4 input videos.

During the development period of HEVC range extension, the palette coding was supported for non-4:4:4 video in the reference software for the ad-hoc group (AhG) on investigation of palette coding mode [3]. Specifically, one dual-palette based method was used to code non-4:4:4 video. In this method, two palette tables are generated, one for the luma component and the other for the two chroma components, by selecting the major colors for luma and chroma components separately. Then, two palette index maps are generated after separately mapping the luma sample and the chroma samples of each pixel into two palette indices (one for luma component and the other for two chroma components). Although it can improve the coding efficiency for non-4:4:4 screen content video, the dual-palette based method requires separate implementation of the palette coding mode for non-4:4:4 video, because it uses two independent palette decoding processes that are operated on the samples with different resolutions. Further, the dual-palette based method needs to maintain two separate palette tables and two separate palette predictors at the decoder. This is more complicated than the existing 4:4:4 palette design which only needs a single palette table and a single palette predictor.

# The proposed method

In this contribution, it is proposed to extend the current 4:4:4 palette design for non-4:4:4 screen content video. The proposed solution maximally reuses the current encoding and decoding logics of the 4:4:4 palette mode. Figure 1 depicts the conceptual decoding flow of the proposed palette coding method for non-4:4:4 video. The intermediate reconstructed samples in 4:4:4 format share exactly the same decoding process as the current decoding process for palette mode as specified in [1]. Then, to output the final reconstructed CU, the conceptual block of chroma sub-sampling is introduced to determine whether to discard a portion of the chroma samples according to the chroma format of the output video. For example, for 4:2:0 output video (as depicted in Figure 2), only the chroma samples located at patterned positions need to be output, and the remaining chroma samples are discarded.

It should be mentioned that the chroma sub-sampling block in Figure 1 is only conceptual. This process actually is combined with the reconstruction process (see proposed text change). 

**Figure 1 Decoding flow of the proposed palette coding method for non-4:4:4 video.**



**Figure 2 Sampling grid of luma and chroma components in 4:2:0 chroma format, where the luma and chroma samples are represented by circles and triangles respectively.**

In the current 4:4:4 palette design [1], the escape colors for all three components need to be signaled through the syntax element *palette\_escape\_val* assuming the 4:4:4 format. For non-4:4:4 video, this signaling method introduces unnecessary overhead for those chroma samples that are not used (i.e. discarded) in the reconstructed CU, for example, the chroma samples located at non-patterned positions in Figure 2. Therefore, in order to reduce the overhead of the palette coding mode for non-4:4:4 video, it is proposed to condition the signaling of *palette\_escape\_val* for chroma components such that only the chroma samples that are used for reconstruction are signaled in bit-stream.

# Simulation results

The proposed method is implemented based on SCM-3.0 and tested for the 4:2:0 screen content sequences as specified in the current common test conditions [4].

Compared to the SCM-3.0 anchor, Table 1 and Table 2 present BD-rate savings of the proposed non-4:4:4 palette method for lossy coding and lossless coding, respectively. As shown in Table 1, for lossy coding, the proposed method provides the average {G/Y, B/Cb, R/Cr} BD-rate savings for AI, RA and LB of {3.2%, 6.3%, 5.5%}, {2.3%, 6.3%, 5.2%} and {1.5%, 5.1%, 4.1%}, respectively. For lossless coding results in Table 2, the corresponding average bit-rate savings for AI, RA and LB are 0.2%, 0.1% and 0%, respectively.

Table 1 BD-rate performance of the proposed non-4:4:4 palette (with conditioned escape color signaling), compared to SCM-3.0 for lossy coding

|  |  |  |  |
| --- | --- | --- | --- |
|  | **All Intra** | | |
|  | G/Y | B/U | R/V |
| Text & graphics with motion, 720p | -3.5% | -3.7% | -4.3% |
| Mixed content, 480p | -0.5% | -2.7% | -2.8% |
| Animation, 768p | -5.3% | -15.1% | -10.6% |
| Average of all sequences | -3.2% | -6.3% | -5.5% |
| Enc Time[%] | 140% | | |
| Dec Time[%] | 102% | | |
|  |  |  |  |
|  | **Random Access** | | |
|  | G/Y | B/U | R/V |
| Text & graphics with motion, 720p | -3.1% | -3.6% | -4.7% |
| Mixed content, 480p | -0.2% | -1.3% | -1.5% |
| Animation, 768p | -2.9% | -16.7% | -10.0% |
| Average of all sequences | -2.3% | -6.3% | -5.2% |
| Enc Time[%] | 121% | | |
| Dec Time[%] | 96% | | |
|  |  |  |  |
|  | **Low delay B** | | |
|  | G/Y | B/U | R/V |
| Text & graphics with motion, 720p | -2.2% | -3.7% | -4.9% |
| Mixed content, 480p | 0.1% | -0.1% | -0.6% |
| Animation, 768p | -1.6% | -12.7% | -6.0% |
| Average of all sequences | -1.5% | -5.1% | -4.1% |
| Enc Time[%] | 117% | | |
| Dec Time[%] | 96% | | |

Table 2 Bit-rate performance of the proposed non-4:4:4 palette (with conditioned escape color signaling), compared to SCM-3.0 for lossless coding

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **All Intra** | | | |
|  | Bit-rate change (Total) | Bit-rate change (Average) | Bit-rate change (Min) | Bit-rate change (Max) |
|  |
| Text & graphics with motion, 720p | -0.1% | -0.1% | -0.3% | 0.0% |
| Mixed content, 480p | -0.1% | -0.1% | -0.1% | -0.1% |
| Animation, 768p | -0.5% | -0.5% | -0.5% | -0.5% |
| Average of all sequences | -0.2% | -0.2% | -0.5% | 0.0% |
| Enc Time[%] | 160% | | | |
| Dec Time[%] | 117% | | | |
|  |  |  |  |  |
|  | **Random Access** | | | |
|  | Bit-rate change (Total) | Bit-rate change (Average) | Bit-rate change (Min) | Bit-rate change (Max) |
|  |
| Text & graphics with motion, 720p | -0.1% | -0.2% | -0.3% | -0.1% |
| Mixed content, 480p | 0.0% | 0.0% | 0.0% | 0.0% |
| Animation, 768p | 0.0% | 0.0% | 0.0% | 0.0% |
| Average of all sequences | 0.0% | -0.1% | -0.3% | 0.0% |
| Enc Time[%] | 127% | | | |
| Dec Time[%] | 110% | | | |
|  |  |  |  |  |
|  | **Low Delay B** | | | |
|  | Bit-rate change (Total) | Bit-rate change (Average) | Bit-rate change (Min) | Bit-rate change (Max) |
|  |
| Text & graphics with motion, 720p | -0.1% | 0.0% | -0.1% | 0.0% |
| Mixed content, 480p | 0.0% | 0.0% | 0.0% | 0.0% |
| Animation, 768p | 0.0% | 0.0% | 0.0% | 0.0% |
| Average of all sequences | 0.0% | 0.0% | -0.1% | 0.0% |
| Enc Time[%] | 119% | | | |
| Dec Time[%] | 104% | | | |

# Proposed specification changes

All changes are highlighted.

#### 7.3.8.8 Palette syntax

|  |  |
| --- | --- |
| palette\_coding( x0, y0, nCbS ) { | Descriptor |
| …… |  |
| scanPos = 0 |  |
| while( scanPos < nCbS \* nCbS ) { |  |
| xC = x0 + travScan[ scanPos ][ 0 ] |  |
| yC = y0 + travScan[ scanPos ][ 1 ] |  |
| runPos = 0 |  |
| while ( runPos < = paletteRun ) { |  |
| …… |  |
| for( cIdx = 0; cIdx < 3; cIdx++ ) { |  |
| if( ( cIdx == 0 ) || ( ChromaArrayType == 1 && (xR&1 == 0) && (yR&1 == 0) ) |  |
| || ( ChromaArrayType == 2 && (xR&1 == 0) ) || ChromaArrayType == 3 ) { |  |
| **palette\_escape\_val** | ae(v) |
| PaletteEscapeVal[ cIdx ][ xR ][ yR ] = palette\_escape\_val |  |
| } |  |
| } |  |
| …… |  |
| runPos++ |  |
| scanPos++ |  |
| } |  |
| } |  |
| } |  |

#### 8.4.1 General decoding process for coding units coded in intra prediction mode

Inputs to this process are:

– a location ( xCb, yCb ) specifying the top-left sample of the current block relative to the top-left sample of the current picture,

– a variable cIdx specifying the colour component of the current block,

– a variable nCbS specifying the size of the current block,

– An array paletteSampleMode[ xCb + x ][ yCb + y ], with x = 0..nCbS − 1 and  y = 0..nCbS − 1, specifying the method of palette index derivation for each sample in the current block

– The array paletteIndexMap[ xCb + x ][ yCb + y ], with x = 0..nCbS − 1,  y = 0..nCbS − 1, specifying the palette index for each sample in the current block, and

– The array paletteEscapeVal[cIdx][ xCb + x ][ yCb + y ], with x = 0..nCbS − 1 and  y = 0..nCbS − 1, specifying the escape value (possibly quantized) for the cIdx-th component of each sample in the current block for which the paletteSampleMode[ xCb + x ][ yCb + y ] is equal to ESCAPE\_MODE.

Depending on the value of the color component cIdx, the variable nSubWidth and nSubHeight are derived as follows:

– If cIdx is equal to 0, nSubWidth is set to 1 and nSubHeight is set to 1.

– Otherwise, nSubWidth is set to SubWidthC and nSubHeight is set to SubHeightC.

Output of this process is an array recSamples[ x ][ y ], with x = 0..nCbS/nSubWidth − 1,  y = 0..nCbS/nSubHeight − 1, specifying reconstructed sample values for the block.

Depending on the value of the colour component cIdx, the following assignments are made:

– If cIdx is equal to 0, the function clipCidx1 corresponds to Clip1Y.

– Otherwise, if cIdx is equal to 1, the function clipCidx1 corresponds to Clip1C.

– Otherwise (cIdx is equal to 2), the function clipCidx1 corresponds to Clip1C.

The variables PredictorPaletteSize, PreviousPaletteSize and PredictorPaletteEntries are derived as follows:

– The initialization process as specified in subclause 9.3.2 is invoked when starting the parsing of one or more of the following:

– the slice segment data syntax specified in subclause 7.3.8.1

– the coding tree unit syntax specified in subclause 7.3.8.2 and the coding tree unit is the first coding tree unit in a tile

– the coding tree unit syntax specified in subclause 7.3.8.2, entropy\_coding\_sync\_enabled\_flag is equal to 1, and the associated luma coding tree block is the first luma coding tree block in a coding tree unit row

– Otherwise, PredictorPaletteSize, PreviousPaletteSize, and PredictorPaletteEntries updated after decoding the previous coding unit with palette\_mode\_flag[ x0 ][ y0 ] equal to 1 are used.

The (nCbS/nSubWidth x nCbS/nSubHeight) block of the reconstructed sample array recSamples at location ( xCb/nSubWidth, yCb/nSubHeight ) is derived as follows:

For x = 0..nCbS/nSubWidth − 1,  y = 0..nCbS/nSubHeight − 1, recSample[ cIdx ][ yCb/nSubHeight + y ][ xCb/nSubWidth + x ] is set as follows:

– If paletteSampleMode[ xCb + x\*nSubWidth ][ yCb + y\*nSubHeight ] is not equal to ESCAPE\_MODE, the following applies:

recSample[ cIdx ][ xCb/nSubWidth + x ][ yCb/nSubHeight + y ] = currentPaletteEntries[ cIdx ][ paletteIndexMap[ xCb + x\*nSubWidth ][ yCb + y\*nSubHeight ] ],

– Otherwise, if cu\_transquant\_bypass\_flag is true, the following applies:

recSample[ cIdx ][ xCb/nSubWidth + x ][ yCb/nSubHeight + y ] = paletteEscapeVal[ cIdx ][ xCb + x\*nSubWidth ][ yCb + y\*nSubHeight ],

– Otherwise (paletteSampleMode[ xCb + x\*nSubWidth ][ yCb + y\*nSubHeight ] is equal to ESCAPE\_MODE and cu\_transquant\_bypass\_flag is false), the following ordered steps apply:

1. The derivation process for quantization parameters as specified in subclause 8.6.1 is invoked with the location ( xCb, yCb ) specifying the top-left sample of the current block relative to the top-left sample of the current picture.
2. The quantization parameter qP is derived as follows:

– If cIdx is equal to 0,

qP = max( 0, Qp′Y ) (8‑72)

– Otherwise, if cIdx is equal to 1,

qP = max( 0, Qp′Cb ) (8‑73)

– Otherwise (cIdx is equal to 2),

qP = max(0, Qp′Cr ) (8‑74)

1. The variables bitDepth and bdShift1 are derived as follows:

bitDepth = ( cIdx = = 0 ) ? BitDepthY : BitDepthC  (8‑75)

bdShift1 = Max( 20 − bitDepth, extended\_precision\_processing\_flag ? 11 : 0 ) (8‑76)

1. The variables log2TransformRange, bdShift2, coeffMin and coeffMax are derived as follows:

– If cIdx is equal to 0,

log2TransformRange = extended\_precision\_processing\_flag ? Max( 15, BitDepthY + 6 ) : 15 (8‑77)

bdShift2 = BitDepthY + Log2( nTbS ) + 10 − log2TransformRange (8‑78)

coeffMin = CoeffMinY (8‑79)

coeffMax = CoeffMaxY (8‑80)

– Otherwise,

log2TransformRange = extended\_precision\_processing\_flag ? Max( 15, BitDepthC + 6 ) : 15 (8‑81)

bdShift2 = BitDepthC + Log2( nTbS ) + 10 − log2TransformRange (8‑82)

coeffMin = CoeffMinC (8‑83)

coeffMax = CoeffMaxC (8‑84)

1. The list levelScale[ ] is specified as levelScale[ k ] = { 40, 45, 51, 57, 64, 72 } with k = 0..5.
2. The following applies

recSamples[ xCb/nSubWidth + x ][ yCb/nSubHeight + y ] =  
Clip3( coeffMin, coeffMax, ( ( palette\_escape\_val[ cIdx ][ xCb + x\*nSubWidth ][ yCb + y\*nSubHeight ]\* 16 \*   
 levelScale[ qP%6 ]  <<  (qP / 6 ) ) + ( 1  <<  ( bdShift2 − 1 ) ) )  >>  bdShift2 ) (8‑63)

1. The reconstructed sample value recSamples[ xCb + x ][ yCb + y ] is modified as follows:

recSamples[ xCb/nSubWidth + x ][ yCb/nSubHeight + y ] = ( recSamples[ xCb/nSubWidth + x ][ yCb/nSubHeight + y ]   
 + ( 1  <<  ( bdShift2 − 1 ) ) ) >> bdShift2 (8‑63)

The variable PreviousPaletteSize is set equal to the CurrentPaletteSize. Variables PredictorPaletteSize and PredictorPaletteEntries are modified as follows:

for( i = 0; i < currentPaletteSize; i++ )   
 for( cIdx = 0; cIdx < 3; cIdx++ )   
 newPredictorPaletteEntries[ cIdx ][ i ] = CurrentPaletteEntries[ cIdx ][ i ]  
newPredictorPaletteSize = currentPaletteSize   
for( i = 0; i < PredictorPaletteSize && newPredictorPaletteSize < palette\_max\_predictor\_size; i++ )  
 if( PalettePredictorEntryReuseFlag[ i ] **!** **=** 1 ) {  
 for( cIdx = 0; cIdx < 3; cIdx++ ) (8‑63)  
 newPredictorPaletteEntries[ cIdx ][ newPredictorPaletteSize ] = PredictorPaletteEntries[ cIdx ][ i ]  
 newPredictorPaletteSize ++   
 }  
for( cIdx = 0; cIdx < 3; cIdx++ )   
 for( i = 0; i < newPredictorPaletteSize; i++ )   
 PredictorPaletteEntries[ cIdx ][ i ] = newPredictorPaletteEntries[ cIdx ][ i ]  
PredictorPaletteSize = newPredictorPaletteSize

# References

1. R. Joshi, J. Xu, HEVC Screen Content Coding Draft Text 2, JCTVC-S1005, Oct. 2014, Strasbourg, France.
2. J. Sole, M. Karczewicz, R. Joshi, V. Seregin, K. Rapaka, AhG6: Inclusion of 4:2:0 Screen Content in the HEVC SCC Common Test Conditions, JCTVC-S0042, Oct. 2014, Strasbourg, France.
3. W. Pu, X. Guo, P. Onno, P. Lai, J. Xu, Suggested Software for The AHG on Investigation of Palette Mode Coding Tools, JCTVC-P0303, Jan. 2014, San Jose, USA.
4. H. Yu, R. Cohen, K. Rapaka, J. Xu, Common Test Conditions for Screen Content Coding, JCTVC-S1015, Oct. 2014, Strasbourg, France.

# Patent rights declaration(s)

**InterDigital Communications LLC may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**