|  |  |
| --- | --- |
| **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-T0119 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Non-CE1: improved palette run-length coding with palette flipping** | | |
| *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

At Strasbourg meeting, CE1 Test B.1 was established to study the performance of the modified run coding method by sending one additional “run-to-the-end” flag to indicate that one run will continue to the end of the block. In this contribution, one improvement method is proposed on top of CE1 Test B.1 by adding one CU-level flag to indicate whether or not to flip the vertical scanning order, i.e., to scan the palette-coded CU from bottom to top. In addition, one encoder method is also proposed to improve the selection between index mode and copy-above mode.

Experimental results show that compared to SCM-3.0 anchor, the proposed method reportedly provides the average {G/Y, B/Cb, R/Cr} BD-rate savings for AI, RA and LB of {0.7%, 0.7%, 0.8%}, {0.4%, 0.5%, 0.6%} and {0.3%, 0.5%, 0.4%} for the category *text & graphics with motion, 1080p & 720p* in both RGB and YCbCr color formats respectively, without noticeable encoding and decoding complexity increase.

# Introduction

According to the current palette design in HEVC screen content coding (SCC) specification draft [1], the last run in one palette-coded CU is always equal to the distance from the leading pixel of the last run to the end of the CU. Therefore, instead of signaling the value of the last run itself, one “run-to-the-end” flag was proposed in [2] to indicate one run (either index mode or copy-above mode) will continue to the end of the current CU. The method is included as Test B.1 in the Core Experiment 1 (CE1) on improvements of palette mode [3].

Although the method in [2] can improve the efficiency of run coding, this method can only be applied to the last run of one palette-coded CU. However, for a palette coded CUs, it has been observed that often a large run is located at the beginning of the block, instead of at the end of the block. Figure 1 (a) shows one example for such case, where the first run is 22 and the last run is 3.

 

**(a) (b)**

**Figure 1 Example palette index map of one 8x8 palette-coded CU. (a) the original palette index map; (b) the flipped palette index map.**

# Proposed methods

## Palette flipping flag

As shown in Figure 1 (a), it is possible that the first run of one palette-coded CU is much larger than the last run. In such case, the “run-to-the-end” flag as proposed in [2] can only be applied to code the small run, which may not provide much bit savings to the run coding. However, for the same case, the run coding efficiency can be improved by scanning the palette index map in the reversed vertical direction, that is, from bottom to top; in this way, as shown in Figure 1 (b), the large run at the beginning of the CU before flipping will be repositioned to the end of the CU after flipping, to which the existing “run-to-the-end” flag could be efficiently applied. Therefore, to maximize the coding gain provided by the “run-to-the-end” flag, it is proposed to signal one additional syntax element *palette\_flipping\_flag* at the CU-level for palette-coded CUs. If the flag is equal to 1, the samples of the CU are flipped vertically before output; otherwise, the samples of the CU are output directly.

## Improved optimization method on palette index coding mode selection (encoder-only)

For each index position in a palette-coded CU, the encoder optimization method in the SCM-3.0 derives the largest possible runs for both index mode and copy-above mode in an aggressive manner and compare the average per-position bit costs of two modes; then, the mode with smaller average bit usage will be selected. However, this method, which is based on best-effort run derivation, may not be optimal for the cases when an index mode is followed by a copy-above mode. In such cases, the overall average may be further improved if different combinations of index run and copy-above run are tested. For example, if reducing the index run will allow a longer copy-above run to follow, then the overall average of bits per position may be reduced. Therefore, in order to enhance the performance of palette index coding, an improved encoder method is proposed to determine the optimal mode and the corresponding run by jointly considering the combination of an index mode run followed by a copy-above mode run. Specifically, for each possible run of index mode at one tested position, the proposed encoder method attempts to derive another copy-above run after the index run. If the copy-above run is enabled, the corresponding average per-position bit cost of the combined mode will be calculated and tested by the encoder. Additionally, to reduce the encoding complexity, the above search process for the optimal combination of index run and copy-above run will be early terminated when it is found that the average per-position bit cost of the combined mode starts to increase.

# Simulation results

The proposed methods are implemented based on SCM-3.0 and tested under the common test conditions as specified in [4].

## The comparison between the proposed methods and SCM-3.0

Table 1 and Table 2 present BD-rate and bit-rate performances of the proposed methods for lossy coding and lossless coding, respectively, compared to the SCM-3.0 anchor. As shown in Table 1, for lossy coding, the proposed methods provide the average {G/Y, B/Cb, R/Cr} BD-rate savings for AI, RA and LB of {0.7%, 0.7%, 0.8%}, {0.4%, 0.5%, 0.6%} and {0.3%, 0.5%, 0.4%}, respectively, for the sequences in the categories *text & graphics with motion, 1080p & 720p* in both RGB and YCbCr color formats. For lossless coding results in Table 2, the corresponding average bit-rate savings for AI, RA and LB are 0.3%, 0.2% and 0.1%, respectively. Additionally, as shown in Table 1 and Table 2, the differences of both the encoding and decoding time between the proposed method and SCM-3.0 anchor are negligible.

Table 1 BD-rate performance of the proposed methods, compared to SCM-3.0 for lossy coding

|  |  |  |  |
| --- | --- | --- | --- |
|  | **All Intra** | | |
|  | G/Y | B/U | R/V |
| RGB, text & graphics with motion, 1080p & 720p | -0.6% | -0.7% | -0.6% |
| RGB, mixed content, 1440p & 1080p | -0.2% | -0.2% | -0.2% |
| RGB, Animation, 720p | 0.0% | 0.0% | 0.0% |
| RGB, camera captured, 1080p | 0.0% | 0.0% | 0.0% |
| YUV, text & graphics with motion, 1080p & 720p | -0.7% | -0.8% | -0.9% |
| YUV, mixed content, 1440p & 1080p | -0.3% | -0.6% | -0.6% |
| YUV, Animation, 720p | 0.0% | -0.2% | -0.1% |
| YUV, camera captured, 1080p | 0.0% | 0.0% | 0.0% |
| Enc Time[%] | 102% | | |
| Dec Time[%] | 100% | | |
|  |  |  |  |
|  | **Random Access** | | |
|  | G/Y | B/U | R/V |
| RGB, text & graphics with motion, 1080p & 720p | -0.4% | -0.5% | -0.4% |
| RGB, mixed content, 1440p & 1080p | -0.1% | -0.1% | -0.2% |
| RGB, Animation, 720p | 0.0% | 0.0% | 0.0% |
| RGB, camera captured, 1080p | 0.0% | 0.0% | 0.0% |
| YUV, text & graphics with motion, 1080p & 720p | -0.4% | -0.6% | -0.7% |
| YUV, mixed content, 1440p & 1080p | -0.2% | -0.3% | -0.5% |
| YUV, Animation, 720p | -0.1% | -0.1% | 0.1% |
| YUV, camera captured, 1080p | 0.0% | 0.0% | 0.0% |
| Enc Time[%] | 102% | | |
| Dec Time[%] | 99% | | |
|  |  |  |  |
|  | **Low delay B** | | |
|  | G/Y | B/U | R/V |
| RGB, text & graphics with motion, 1080p & 720p | -0.3% | -0.4% | -0.4% |
| RGB, mixed content, 1440p & 1080p | -0.3% | -0.2% | -0.3% |
| RGB, Animation, 720p | 0.0% | -0.1% | -0.1% |
| RGB, camera captured, 1080p | 0.0% | 0.0% | 0.0% |
| YUV, text & graphics with motion, 1080p & 720p | -0.4% | -0.5% | -0.5% |
| YUV, mixed content, 1440p & 1080p | -0.1% | -0.3% | -0.3% |
| YUV, Animation, 720p | 0.0% | -0.1% | 0.0% |
| YUV, camera captured, 1080p | 0.0% | 0.1% | -0.2% |
| Enc Time[%] | 101% | | |
| Dec Time[%] | 101% | | |
|  |  |  |  |

Table 2 Bit-rate performance of the proposed methods, 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) |
|  |
| RGB, text & graphics with motion, 1080p & 720p | -0.2% | -0.3% | -0.6% | -0.1% |
| RGB, mixed content, 1440p & 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| RGB, Animation, 720p | 0.0% | 0.0% | 0.0% | 0.0% |
| RGB, camera captured, 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| YUV, text & graphics with motion, 1080p & 720p | -0.3% | -0.3% | -0.7% | -0.1% |
| YUV, mixed content, 1440p & 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| YUV, Animation, 720p | 0.0% | 0.0% | 0.0% | 0.0% |
| YUV, camera captured, 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| Enc Time[%] | 98% | | | |
| Dec Time[%] | 94% | | | |
|  |  |  |  |  |
|  | **Random Access** | | | |
|  | Bit-rate change (Total) | Bit-rate change (Average) | Bit-rate change (Min) | Bit-rate change (Max) |
|  |
| RGB, text & graphics with motion, 1080p & 720p | -0.1% | -0.2% | -0.5% | 0.0% |
| RGB, mixed content, 1440p & 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| RGB, Animation, 720p | 0.0% | 0.0% | 0.0% | 0.0% |
| RGB, camera captured, 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| YUV, text & graphics with motion, 1080p & 720p | -0.1% | -0.2% | -0.5% | 0.0% |
| YUV, mixed content, 1440p & 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| YUV, Animation, 720p | 0.0% | 0.0% | 0.0% | 0.0% |
| YUV, camera captured, 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| Enc Time[%] | 95% | | | |
| Dec Time[%] | 99% | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  | **Low Delay B** | | | |
|  | Bit-rate change (Total) | Bit-rate change (Average) | Bit-rate change (Min) | Bit-rate change (Max) |
|  |
| RGB, text & graphics with motion, 1080p & 720p | -0.1% | -0.1% | -0.3% | 0.0% |
| RGB, mixed content, 1440p & 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| RGB, Animation, 720p | 0.0% | 0.0% | 0.0% | 0.0% |
| RGB, camera captured, 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| YUV, text & graphics with motion, 1080p & 720p | -0.1% | -0.1% | -0.3% | 0.0% |
| YUV, mixed content, 1440p & 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| YUV, Animation, 720p | 0.0% | 0.0% | 0.0% | 0.0% |
| YUV, camera captured, 1080p | 0.0% | 0.0% | 0.0% | 0.0% |
| Enc Time[%] | 96% | | | |
| Dec Time[%] | 100% | | | |
|  |  |  |  |  |

# Proposed specification changes

#### Text changes for the method in CE1 Test B1

#### Text changes for the proposed palette flipping

#### 7.3.8.8 Palette syntax

|  |  |
| --- | --- |
| palette\_coding( x0, y0, nCbS ) { | Descriptor |
| …… |  |
| if( indexMax > 0){ |  |
| **palette\_transpose\_flag** | ae(v) |
| **palette\_flipping\_flag** | ae(v) |
| } |  |
| scanPos = 0 |  |
| while( scanPos < nCbS \* nCbS ) { |  |
| …… |  |
| if( maxPaletteRun > 0 ) { |  |
| **palette\_run\_msb\_id\_plus1** | ae(v) |
| if( palette\_run\_msb\_id\_plus1 > 1 ) { |  |
| if( scanPos && (1<< palette\_run \_msb\_id\_plus1) > MaxPaletteRun ) |  |
| **palette\_run\_to\_end\_flag** | ae(v) |
| if( !palette\_last\_run\_flag **)** |  |
| **palette\_run\_refinement\_bits** | ae(v) |
| } |  |
| } |  |
| …… |  |
| } |  |
| } |  |

#### 7.4.9.6 Palette mode semantics

**palette\_flipping\_flag** equal to 1 specifies that the flipping process is applied to the current coding unit before reconstruction. palette\_flipping\_flag equal to 0 specifies that the flipping process is not applied to the current coding unit. When not present, the value of palette\_flipping\_flag is inferred to be 0.

**palette\_run\_to\_end\_flag** equal to 1 specifies that paletteRun is equal to MaxPaletteRun. palette\_ run\_to\_end\_flag equal to 0 specifies that paletteRun is not equal to MaxPaletteRun.

When palette\_run\_to\_end\_flag is not present, it is inferred to be equal to 0.

The variable paletteRun is derived as follows:

* If indexMax is greater than 0
* If palette\_run**\_**msb\_id\_plus1 is greater than 1,

If palette\_run\_to\_end\_flag is equal to 1,

paletteRun = MaxPaletteRun (7‑80)

Else

paletteRun = ( 1 << ( palette\_run\_msb\_id\_plus1 − 1 ) ) + palette\_run**\_**refinement\_bits (7‑80)

* Otherwise ( palette\_run\_msb\_id\_plus1 equal to 1 ) paletteRun is set to (palette\_run\_msb\_id\_plus1 – 1).
* Otherwise, paletteRun is set to ( nCbS \* nCbS – 1 ).

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

Inputs to this process are:

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

– a variable log2CbSize specifying the size of the current luma coding block.

Output of this process is a modified reconstructed picture before deblocking filtering.

……

– If pcm\_flag[ xCb ][ yCb ] is equal to 1, the reconstructed picture is modified as follows:

SL[ xCb + i ][ yCb + j ] =   
 pcm\_sample\_luma[ ( nCbS \* j ) + i ]  <<  ( BitDepthY − PcmBitDepthY ), with i, j = 0..nCbS − 1 (8‑12)

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0), if palette\_mode\_flag[ xCb ][ yCb ] is equal to 1, the following order steps apply:

1. The decoding process for palette intra blocks as specified in subclause 8.4.5.2.8 is invoked with the luma location ( xCb, yCb ), nCbS, the variable cIdx set equal to 0, the array paletteSampleMode, the palette indices array paletteIndexMap, and the array of (possibly quantized) escape values paletteEscapeVal as inputs, and the output is an nCbS x nCbS array of reconstructed -palette sample values, recSamples[ x ][ y ], x, y = 0..nCbS − 1
2. The reconstructed picture is modified as follows:

– If palette\_transpose\_flag is true,

* + If palette\_flipping\_flag is true

SL[ yCb + y ][ xCb + x ] = recSamples[ x ][ nCbS- y ]

* + Otherwise (palette\_flipping\_flag is false)

SL[ yCb + y ][ xCb + x ] = recSamples[ x ][ y ]

– Otherwise (palette\_transpose\_flag is false)

* + If palette\_flipping\_flag is true

SL[ xCb + x ][ yCb + y ] = recSamples[ x ][nCbS - y ]

* + Otherwise (palette\_flipping\_flag is false)

SL[ xCb + x ][ yCb + y ] = recSamples[ x ][ y ].

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0, palette\_mode\_flag[ xCb ][ yCb ] is equal to 0 ), if IntraSplitFlag is equal to 0, the following ordered steps apply:

**……**

When ChromaArrayType is not equal to 0, the following applies.

The variable log2CbSizeC is set equal to log2CbSize − ( ChromaArrayType  = =  3 ? 0 : 1 ).

Depending on the values of pcm\_flag[ xCb ][ yCb ] and IntraSplitFlag, the decoding process for chroma samples is specified as follows:

– If pcm\_flag[ xCb ][ yCb ] is equal to 1, the reconstructed picture is modified as follows:

SCb[ xCb / SubWidthC + i ][ yCb / SubHeightC + j ] =   
 pcm\_sample\_chroma[ ( nCbS / SubWidthC \* j ) + i ]  <<  ( BitDepthC − PcmBitDepthC ),  
 with i = 0..nCbS / SubWidthC − 1, and j = 0..nCbS / SubHeightC − 1 (8‑13)

SCr[ xCb / SubWidthC + i ][ yCb / SubHeightC + j ] =   
 pcm\_sample\_chroma[ ( nCbS / SubWidthC \* ( j + nCbS / SubHeightC ) ) + i ]  <<

( BitDepthC − PcmBitDepthC ),  
 with i = 0..nCbS / SubWidthC − 1, and j = 0..nCbS / SubHeightC − 1 (8‑14)

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0), if palette\_mode\_flag[ xCb ][ yCb ] is equal to 1 the following orderd steps apply:

1. The decoding process for palette intra blocks as specified in subclause 8.4.5.2.8 is invoked with the chroma location ( xCb, yCb ), nCbS, the variable cIdx set equal to 1, the array paletteSampleMode, the palette indices array paletteIndexMap, and the array of (possibly quantized) escape values paletteEscapeVal as inputs, and the output is an nCbS x nCbS array of reconstructed palette sample values, recSamples[ x ][ y ], x, y = 0..nCbS − 1.
2. The reconstructed picture is modified as follows:

– If palette\_transpose\_flag is true~~, S~~~~Cb~~~~[ yCb + y ][ xCb + x ] is set equal to recSamples[ x ][ y ],~~

* + If palette\_flipping\_flag is true

SCb[ yCb/ SubHeightC  + y ][ xCb/SubWidthC  + x ] = recSamples[ x ][ nCbS/ SubHeightC - y ]

* + Otherwise (palette\_flipping\_flag is false)

SCb[ yCb/SubHeightC + y ][ xCb/SubWidthC + x ] = recSamples[ x ][ y ]

– Otherwise (palette\_transpose\_flag is false), ~~S~~~~Cb~~~~[ xCb + x ][ yCb + y ] is set equal to recSamples[ x ][ y~~].

* + If palette\_flipping\_flag is true

SCb[ xCb/SubWidthC + x ] [ yCb/ SubHeightC + y ] = recSamples[ x ][ nCbS/ SubHeightC - y ]

* + Otherwise (palette\_flipping\_flag is false)

SCb[ xCb/SubWidthC + x ] [ yCb/ SubHeightC + y ] = recSamples[ x ][ y ]

1. The decoding process for palette intra blocks as specified in subclause 8.4.5.2.8 is invoked with the chroma location ( xCb, yCb ), nCbS, the variable cIdx set equal to 2, the array paletteSampleMode, the palette indices array paletteIndexMap, and the array of (possibly quantized) escape values paletteEscapeVal as inputs, and the output is an nCbS x nCbS array of reconstructed palette sample values, recSamples[ x ][ y ], x, y = 0..nCbS − 1.
2. The reconstructed picture is modified as follows:

– If palette\_transpose\_flag is true~~, S~~~~Cr~~~~[ yCb + y ][ xCb + x ] is set equal to recSamples[ x ][ y ]~~

* + If palette\_flipping\_flag is true

SCr[ yCb/ SubHeightC  + y ][ xCb/SubWidthC  + x ] = recSamples[ x ][ nCbS/ SubHeightC - y ]

* + Otherwise (palette\_flipping\_flag is false)

SCr[ yCb/SubHeightC + y ][ xCb/SubWidthC + x ] = recSamples[ x ][ y ]

– Otherwise (palette\_transpose\_flag is false), ~~S~~~~Cr~~~~[ xCb + x ][ yCb + y ] is set equal to recSamples[ x ][ y ]~~

* + If palette\_flipping\_flag is true

SCr[ xCb/SubWidthC + x ] [ yCb/ SubHeightC + y ] = recSamples[ x ][ nCbS/ SubHeightC - y ]

* + Otherwise (palette\_flipping\_flag is false)

SCr[ xCb/SubWidthC + x ] [ yCb/ SubHeightC + y ] = recSamples[ x ][ y ]

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0, palette\_mode\_flag[ xCb ][ yCb ] is equal to 0 ), if IntraSplitFlag is equal to 0 or ChromaArrayType is not equal to 3, the following ordered steps apply:

**……**

#### 9.3.3.1 General

Table 9‑38 – Syntax elements and associated binarizations

|  |  |  |  |
| --- | --- | --- | --- |
| palette\_coding( ) | … | … | … |
| palette\_flipping\_flag | FL | cMax = 1 |
| … | … | … |
| palette\_run\_msb\_id\_plus1 | TR | cMax = Floor( Log2((MaxPaletteRun ) ) + 1, cRiceParam = 0 |
| palette\_run\_to\_end\_flag | FL | cMax = 1 |
| palette\_run\_refinement\_bits | TB | cMax = ((1<< palette\_run \_msb\_id\_plus1) >= MaxPaletteRun) ? MaxPaletteRun – (1 <<( palette\_run\_msb\_id\_plus1-1)) – 1: (1<< (palette\_run \_msb\_id\_plus1 – 1)) – 1 |
| … | … | … |

#### 9.4.1.2.1 General

Table 9‑43 – Assignment of ctxInc to syntax elements with context coded bins

| **Syntax element** | **binIdx** | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **>= 5** |
| … | … | … | … | … | … | … |
| palette\_flipping\_flag | 0 | na | na | na | na | na |
| … | … | … | … | … | … | … |
| palette\_run\_msb\_id\_plus1 | (subclause 9.4.1.2.8) | | | | | |
| palette\_run\_to\_end\_flag | bypass | na | na | na | na | na |
| … | … | … | … | … | … | … |

# References

1. R. Joshi, J. Xu, HEVC Screen Content Coding Draft Text 2, JCTVC-S1005, Oct. 2014, Strasbourg, France.

1. [W. Pu](mailto:wpu@qti.qualcomm.com), [R. Joshi](mailto:rajanj@qti.qualcomm.com), M. Karczewicz, F. Zou, V. Seregin, J. Sole, Non-CE6: Improvement on Palette Rn Coding, JCTVC-S0111, Oct. 2014, Strasbourg, France.
2. P. Lai, P. Onno, R. Cohen, V. Seregin, X. Xiu, Z. Ma, Description of Core Experiment 1 (CE1): Improvements of Palette Mode, JCTVC-S1101, Oct. 2014, Strasbourg, France.
3. 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).**