Palette-based Coding in the Screen Content Coding Extension of the HEVC Standard

Xiaoyu Xiu, Yuwen He[*], Rajan Joshi, Marta Karczewicz[+], Patrice Onno, Christophe Gisquet and Guillaume Laroche[#]

[*]*InterDigital Communications Inc. San Diego, CA, USA*

[+]*Qualcomm Technologies Inc. San Diego, CA, USA*

[#]*Canon Research CentreFrance Cesson-Sévigné, France*

***Abstract**:* This paper provides a technical overview of palette-based coding that was adopted into the test model for the screen content coding (SCC) extension of High Efficiency Video Coding (HEVC) standard at the 18th JCT-VC meeting. Key techniques that enable the palette mode to deliver significant coding gains for screen contents are highlighted, including palette table generation, palette table coding, and the coding methods for palette indices and escape colors. Proposed and adopted techniques up to the first version of the working draft of HEVC SCC extension and test model SCM-2.0 are presented. Experimental results are provided to evaluate the performance of the palette mode in the SCC extension of HEVC.

1. Introduction

ISO/IEC MPEG and ITU-T VCEG formed the Joint Collaborative Team on Video Coding (JCT-VC) to develop the next generation video coding standard, i.e. High Efficiency Video Coding (HEVC). The first version of HEVC [1] was finalized in January 2013. Although the HEVC standard aims to cover a wide range of applications with various performance and implementation requirements, its primary focus is camera-captured natural video.

Due to the rapid growth of Internet and wireless communication, screen content is an integral part of numerous networked video applications, such as desktop sharing, video conferencing, and mobile media presentation. Compared to camera-captured natural content, screen content such as text and graphics shows distinct properties. For example, screen content is rich in areas containing sharp edges (*e.g.*, detailed structure of characters) that can produce artifacts which impair users' viewing experience or distinctly increase the bit rate when transform based coding approaches are applied. As the properties of screen content video sequences are not fully considered during the development of the first version of HEVC [1], it was shown in [2] that superior coding efficiency can be achieved with additional tools over HEVC for screen content materials. Based on this evidence, a Call for Proposals (CfP) [3] on the extensions of the HEVC for screen content coding (SCC) was issued in January 2014 and the responses were reviewed at the 17th JCT-VC meeting in April 2014. Since then, three coding tools, namely intra block copy [4], adaptive color transform [5], and palette-based coding [6], were adopted into working draft 1 of the HEVC SCC specification [7], and show substantial coding gains for screen content video sequences. This paper describes the palette-based coding technique in the current working draft of HEVC SCC specification as well as some variations that were investigated.

1068-0314/15 $31.00 © 2015 IEEE
DOI 10.1109/DCC.2015.79

The palette-based coding in HEVC SCC draft specifications is comprised of three main components: palette table derivation, palette table coding and palette index coding which are respectively described in Sections 3, 4 and 5. Section 2 provides an overview of the palette-based coding algorithm. Section 6 provides experimental results. Conclusions are provided in Sec. 7. Note that the methods presented in this paper reflect the latest standardization activities which significantly extend the preliminary palette mode coding method reported in [8]. Also, the method described in Section 3 is only one non-normative recommendation on how to derive palette table and not included in the HEVC SCC specification [7] that only specifies the decoding process.

2. Algorithm Overview

A palette-based coding method was introduced in [6] and incorporated into the recursive quad-tree framework of HEVC to effectively improve the coding efficiency of screen content video by considering its specific characteristics. Figure 1 illustrates the encoding process of the palette-based coding method.

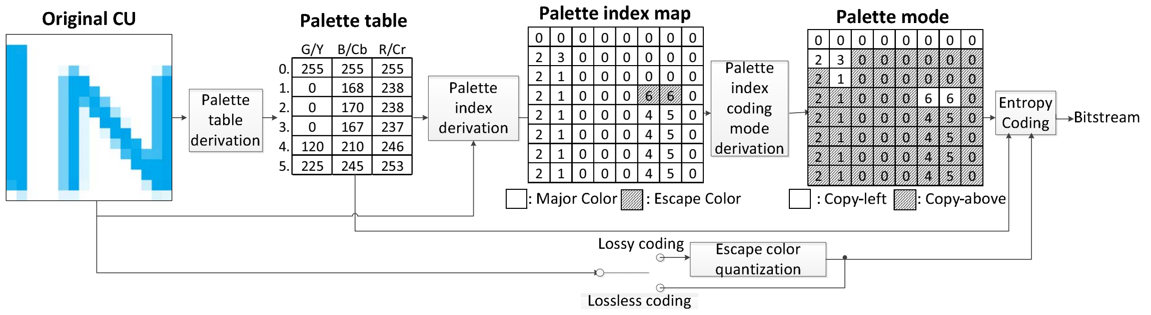


Figure 1: Block diagram of the encoding process of palette-based coding method.

More specifically, for each coding unit (CU) coded with palette mode, a palette table is derived by selecting a set of representative colors from that CU. Then, a palette index map is generated by classifying the pixels of the CU into major colors and escape colors, which are marked as blank blocks and patterned blocks respectively in Fig 1. For the pixels whose colors exist in the palette table, only the indices in the palette are encoded. For the pixels whose colors do not exist in the palette table, their color values are considered as escape colors. The quantized color values (if lossy coding is used) are then directly encoded. To improve the coding efficiency of palette index, two predictive coding modes, *i.e.*, copy-left mode and copy-above mode, are introduced to encode the palette index map by signaling how many consecutive pixels have the same palette index as the current pixel or their respective above neighboring pixels. In the following sections, more detailed descriptions of the key modules of the palette-based coding method in Figure 1 will be presented.

3. Palette Table Derivation

In HEVC SCC Test Model SCM-2.0 [9], the palette table derivation method [10], [11] is used, which consist of: 1) color clustering, 2) major color derivation, and 3) pruning.

In the first step, the color values of the current CU are clustered into K sets using modified K-means method, where K is the palette table size. Denote the original color values in the current CU as $\boldsymbol{c} = \{c_0, c_1, \cdots c_{N-1}\}$, where each color value c_i is a three-dimension vector and N is the total number of pixels in the CU. The color clustering operation aims to partition the colors of the N pixels into $K\,(K \leq N)$ sets $\boldsymbol{S} = \{S_0, S_1, \cdots S_{K-1}\}$ so as to minimize the within-cluster distortion, as formulated as:

$$\underset{S}{arg\,min} \sum_{i=0}^{K-1} \sum_{c \in S_i} \|c - \mu_i\| \tag{1}$$

where μ_i is the centroid of S_i. To solve the optimization problem in (1), a k-mean refinement algorithm is used in [11] to cluster the color values of a CU coded with palette mode.

Given the derived color clusters, the second step determines the major color for each color cluster. In general, the original centroids of the color clusters may be used as the major colors to form the palette table of the current CU. However, as discussed in Sec. 4, each entry of the palette table may be predicted from the corresponding entry of the palette table predictor. Therefore, the original cluster centroids may not always provide the optimal tradeoff between the distortion and the bits required to signal major colors. In order to optimize the rate-distortion (R-D) performance, a Lagrangian R-D comparison is introduced to decide whether the centroid of a color cluster should be replaced by an element from the palette table predictor. Denote the color values in the palette table predictor as $\boldsymbol{u}$. For the i-th color cluster S_i, given the original cluster centroid μ_i, the major color c^* is selected as the one which minimizes the R-D cost as follows:

$$c^* = \underset{c \in \{\mu_i, \boldsymbol{u}\}}{\arg\min}\, D_c + \lambda\, R_c \tag{2}$$

where D_c and R_c denote the total distortion between the color value c and the original color values that are mapped to c, and the bits required to coding the color value respectively, λ is the Lagrangian multiplier.

After the major colors are generated, pruning is performed to remove any duplicated entries such that only unique color values are maintained. In addition, color clusters that contain only one pixel, are also removed from the palette table and regarded as escape colors unless they can be predicted from the palette table predictor.

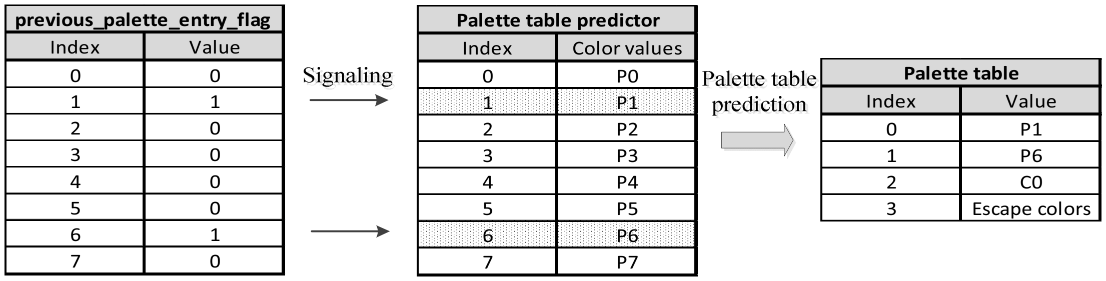
In SCM-2.0 [9], the maximum size of palette table K_{max} is 31. Moreover, if there are escape colors in the current CU, the palette indices $0, 1, \cdots K-1$ are assigned to indicate major colors while the index K is assigned to indicate escape colors.

4. Palette Table Coding

For palette-based coding, palette table should be transmitted to the decoder, which could result in non-negligible signaling overhead. To reduce the overhead of palette table signaling, a palette table prediction scheme is applied to predict the major colors in the current palette table. In SCM-2.0 [9], there are two palette table prediction modes, *i.e.*, explicit palette mode and implicit palette mode (also called palette sharing mode).

4.1. Explicit palette mode

When a CU is coded in explicit palette mode, the palette table is predictively coded using the palette table predictor which may correspond to palette entries from prior CUs coded in palette mode. Additionally, an identification flag named as *previous_palette_entry_flag* is signaled for each element of the palette table predictor to indicate if it is reused in the palette table of the current CU. Given the strong correlations between the colors of neighboring CUs, the major colors that are predicted from the palette table predictor are placed at the beginning of the palette table. Assuming a palette table of size is 3 and a palette table predictor of size is 8, Figure 2 illustrates the palette table generation in the explicit palette mode. In Figure 2, the palette table index $K = 3$ is used to represent escape colors.



previous_palette_entry_flag	
Index	Value
0	0
1	1
2	0
3	0
4	0
5	0
6	1
7	0

Palette table predictor	
Index	Color values
0	P0
1	P1
2	P2
3	P3
4	P4
5	P5
6	P6
7	P7

Palette table	
Index	Value
0	P1
1	P6
2	C0
3	Escape colors

Figure 2: Example of palette table generation in explicit palette mode.

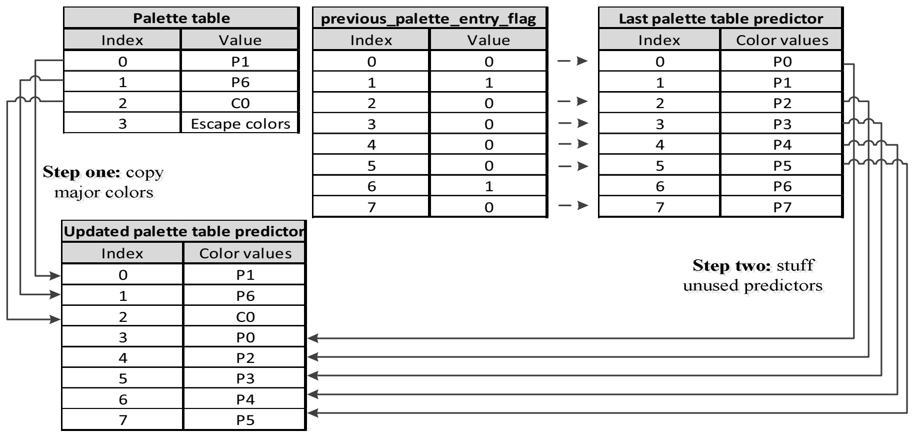
To reduce the overhead of palette table signaling, a two-level hierarchical coding method [12] is applied to code the values of *previous_palette_entry_flag*'s in explicit mode. Specifically, all the reuse flags are divided into groups. Each group contains four flags except the last group which may contain less than four flags. At the group level, two additional syntax elements are signaled for each group: *palette_all_zero_in_group* is a flag indicating if all the flags in the group are all zeros; *palette_last_group* is a flag indicating if the current group is the last group that contains non-zero flag. If *palette_all_zero_in_group* indicates that there is a non-zero flag within a group, four additional flags in that group are further signaled. After sending the reuse flags, for the non-predictable colors of the palette table, the corresponding color values are directly signaled into the bit-stream by a syntax element called *palette_entries*.

4.2. Implicit palette mode

SCM-2.0 [9] includes another mode for signaling the palette table known as the implicit palette mode where the palette table of the current CU is a copy of the palette used for the previously palette-coded CU. It is also referred to as palette sharing mode since it generates multiple CUs sharing the same palette table [13], [14]. More specifically, for each CU coded with palette mode, one syntax element called *palette_share_flag* is signaled to indicate if all the major colors in the current palette table are copied from the previously palette-coded CU. When *palette_share_flag* is equal to 1, the signaling of all the syntax elements related with palette explicit mode in Section 4.1 could be skipped.

4.3. Palette table predictor update

In order to enhance the efficiency of palette table prediction, after one CU is coded with palette mode, a palette predictor stuffing method [13], [15] is used to update palette table predictor. Figure 3 illustrates the update process of palette table predictor using the same example in Figure 2. As shown in Figure 3, the palette table predictor is updated by two main steps: firstly, the major colors of the current CU are added into the palette table predictor; then, the elements in the original palette table predictor that do not exist in the palette table of the current CU are added into the palette table predictor until the palette table predictor reach its size limit. In SCM-2.0 [9], the maximum size of palette table predictor L_{max} is 64.



Palette table

Index	Value
0	P1
1	P6
2	C0
3	Escape colors

previous_palette_entry_flag

Index	Value
0	0
1	1
2	0
3	0
4	0
5	0
6	1
7	0

Last palette table predictor

Index	Color values
0	P0
1	P1
2	P2
3	P3
4	P4
5	P5
6	P6
7	P7

Updated palette table predictor

Index	Color values
0	P1
1	P6
2	C0
3	P0
4	P2
5	P3
6	P4
7	P5

Step one: copy major colors
Step two: stuff unused predictors

Figure 3: Example of palette table predictor update.

5. Palette Index Coding

As shown in Figure 1, a palette index map is generated using the palette table for the pixels in the current CU. The palette-based coding in SCM-2.0 [9] introduces several tools to compress palette index map, which strive to achieve a balance between coding efficiency and practicality of hardware implementation.

5.1. Predictive palette index coding

As shown in Figure 1, there are significant correlations between the palette indices of adjacent pixels. For this reason, run-length based coding techniques are applied to improve the efficiency of palette index coding. Specifically, a flag named *palette_mode* is signaled to select one of the following two modes which reflect different color patterns in the CU.

1. "Copy-left mode": In this mode, the value of a palette index is first signaled through a syntax element *palette_index* and is followed by a non-negative value *palette_run* which indicates the number of the subsequent pixels that have the same palette index as the current pixel.
2. "Copy-above mode": In this mode, the palette index for the current pixel is copied from the pixel directly above it. Only the syntax element *palette_run* is signaled to indicate how many subsequent palette indices are also copied from the respective indices directly above..

In SCM-2.0 [9], two methods were adopted to reduce *palette_mode* signaling cost. First, this flag is not coded for the pixels in the first row of the CU. As there is no above palette index to copy, these pixels can be inferred to be in copy-left mode. Second, if one pixel is right after a series of pixels that are coded in copy-above mode, it is not allow for it to be coded in copy-above mode (otherwise, this pixel could must be coded by simply increasing the run-length of the copy-above mode by 1). Therefore, in this case, the signaling of the *palette_mode* flag is bypassed and the pixel is always inferred to be coded in copy-left mode.

5.1.1 Index coding

For pixels coded in copy-left mode, the palette index needs to be signaled into the bitstream. In SCM-2.0 [9], palette index is binarized using truncated binary code (TBC)

[16]. TBC is a typical binarization method used for a uniform distributed alphabet. It is a more general form of fixed length code (FLC) when the size of the alphabet is not a power of two. Specifically, for one palette index value equals to $pLevel$ with its maximum achievable value denoted as $pMax$, its binarization process can be described as follows.

Let $n = pMax + 1$ and $k = floor\big(log_2(n)\big)$ such that $2^k \leq n < 2^{k+1}$, and $u = 2^{k+1} - n$. $pLevel$ is then binarized as: if $pLevel < u$, the codeword is specified by the binary representation of $pLevel$ with length k; otherwise, the codeword is specified by the binary representation of $pLevel + u$ with length $k + 1$. As an example, Table 1 shows the binarization of palette index when $pMax = 8$. As shown in Table 1, all the TBC codewords of length $k + 1$ are formed by simply appending "0" and "1" to the unassigned codewords of length k.

Table 1 : Example of palette index binarization with $pMax = 8$

Level	**Codeword**	**Level**	**Codeword**
0	000	5	101
1	001	6	110
2	010	7	1110
3	011	8	1111
4	100		

To further improve the efficiency of palette index coding, one redundancy removal method [16], [17], [18] is introduced to reduce the magnitude of the coded levels so that the average length of TBC codewords can be reduced. More specifically, the following conditions are checked before coding each palette index for run mode.

a) If the left neighboring pixel is the end of a copy-left mode, then the current palette index cannot be the same as that of its left neighbor (otherwise, if the two palette indices are the same, they would be coded together in copy-left mode to form a longer run).
b) Similarly, if the left neighboring pixel is the end of a copy-above mode, then the current palette index cannot be the same as that of its above neighbor (otherwise, if the two palette indices are the same, they would be coded together in copy-above mode to form a longer run).

If either of the above two conditions is satisfied, both the level $pLevel$ and its maximum value $pMax$ are reduced by one, which reduces the bits used to signal the palette index of the current pixel.

Additionally, one CU-level flag *palette_escape_val_present_flag* is signaled to indicate if any escape pixel may exist in the current CU or not. In order to reduce the bits used for palette index coding, the maximum input value $pMax$ for the TBC is dependent on the value of *palette_escape_val_present_flag*: if it is equal to 1, $pMax$ is set to palette table size; otherwise, $pMax$ is set to palette table size minus one.

5.1.2 Run length coding

The run coding in SCM-2.0 [9] directly inherits from the coefficient coding method in HEVC [1]. For the palette mode, the binarization of the *palette_run* syntax element is performed according to the Table 2. When the value of the *palette_run* is strictly superior to 2, a prefix value '*111*' is used and the suffix value is binarized using truncated rice (TR) code with the rice parameter $cRiceParam = 3$.

Table 2 : Binarization of palette_run

Palette_run	**Codeword**
0	0
1	10
2	110
>2	Prefix = 111, Suffix = TR code , cRiceParam = 3

5.2. Adaptive index scan

The two-dimensional (2-D) array of the palette indices of one palette-coded CU needs to be firstly converted into a 1-D vector using a scanning pattern before coding palette indices. The aim of scanning is increase the average run-length by grouping pixels with identical palette indices together.. In the original design of palette mode [19], raster scan is used. Raster scan is inefficient in that it tends to break the runs when transitioning from one row to another.. Traverse scan [20] is a variant of raster scan, which scans the rows of the palette index map from left to right and from right to left alternatively. Traverse scan provides better exploitations of the correlations between neighboring palette indices, resulting in a higher efficiency of palette index coding. In SCM-2.0 [9], traverse scan is used to replace raster scan for scanning palette index map.

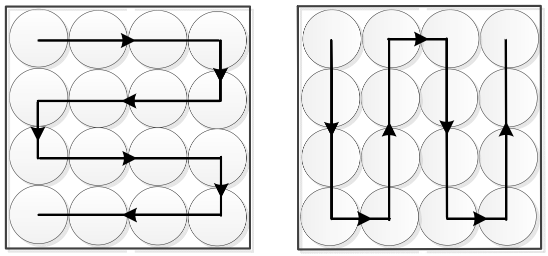


Figure 4: Traverse scan. Left: horizontal traverse scan. Right: vertical traverse scan.

In addition to horizontal traverse scan, vertical traverse scan is also enabled. Specifically, one flag *palette_transpose_flag* is signaled for each CU coded with palette mode. If this flag is equal to 1, the palette indices of the CU are scanned in vertical traverse scan order. Otherwise, the palette indices are scanned in horizontal traverse scan order. Figure 4 illustrates the two traverse scan patterns for 4×4 CU.

5.3. Escape color coding

For each escape color pixel, its color values need to be quantized (if lossy coding is used) and transmitted to decoder. To reduce the overhead of escape color signaling, the TBC used for coding palette index in Sec. 5.1.1 is employed to code the escape color values, where the maximum input value $pMax$ is determined based on the bit-depth of the input video sequence as well as the quantization step size when lossy coding is used.

6. Experimental Results

This section provides experimental data to illustrate the benefits of using palette-based coding for screen content video sequences. All the experiments are conducted using the HEVC SCC Test Model SCM-2.0 where a search over the entire picture is performed for 8x8 and 16x16 blocks coded with intra block copy mode [9]. The performance of palette-based coding is measured by comparing the BD-rates [21] when the palette mode is disabled and enabled. All the conducted experiments follow the common test conditions [22] defined in the scope of the HEVC SCC standard development:

- Lossless mode and lossy mode with four quantization values (22, 27, 32 and 37) are tested.
- Four categories of 26 test sequences in RGB or YCbCr color format are used, namely "*text and graphics with motion*" (TGM), "*mixed content*" (MC), "*animation*" (AN), and "*camera-captured content*" (CC).
- Three different coding structures are evaluated, including all intra (AI), random access (RA), and low delay with B pictures (LB).

Table 3 summarizes the sequence-by-sequence BD-rate savings of enabling palette-based coding under the lossy and lossless configurations. In the results, the anchor corresponds to the HEVC SCC Test Model SCM-2.0 [9] where the palette-based coding is disabled. In Table 3, a positive number means that the palette-based coding has coding gains. For lossy coding, the average BD-rate savings of the palette-based coding are 9.0%, 6.1% and 3.8% for AI, RA, and LB respectively. Regarding lossless coding, the corresponding average bit-rate savings are 6.1%, 3.2%, and 2.2% for AI, RA, and LB respectively. These results show that palette-based coding is effective for the sequences with computer-generated screen contents, i.e. the TGM category (both in 1080p and 720p resolutions). In this category, for lossy coding, the average BD-rate savings even reach 15.3%, 10% and 6.2% for AI, RA, and LB respectively; for lossless coding, the average bit-rate savings are 8.9%, 4.2%, and 3% respectively. These results also confirm that although palette mode is designed for screen content video sequences, it does not affect the coding efficiency for natural contents, i.e. AN and CC categories. The palette-based coding provides higher coding gains in AI configuration than in RA and LB configurations. This is due to the relatively larger percentage of CUs coded with the palette mode in AI case where there is no inter prediction. Additionally, it should be noticed that the palette coding mode is directly competing with the intra block copy mode whose performance is closely related with the size of the search range. Therefore, the palette-based coding is expected to provide more coding gains when a local search is applied for intra block mode.

7. Conclusion

The palette-based coding technique has been adopted into the test model and the working draft for HEVC screen content coding extension. In this paper, the design of the palette coding mode in HEVC SCC working draft specification version 1 is described in detail. It is composed of three components: palette table derivation, palette table coding, and palette index coding. The key techniques in the current palette coding design, which provide high coding efficiency for screen content video, are highlighted. Experimental results are provided which show that for typical screen content video sequences, the palette-based coding provides significant coding performance improvement for computer-generated screen contents without penalty on the coding efficiency for natural contents.

It is acknowledged that this paper only describes the palette coding techniques adopted into the first working draft of HEVC SCC extension [7], which may not be the final design. Multiple core experiments (CE) and ad-hoc groups (AHG) have been established to identify further coding performance improvement and complexity reduction over the current design. For more information, please refer to [23].

Table 3: Summarized luma BD-rate reductions of palette coding under different coding configurations for both lossy coding and lossless coding

Color format	Category	Sequences	Lossy			Lossless		
			AI	RA	LB	AI	RA	LB
RGB	TGM 1080p	FlyingGraphics	13.4%	6.9%	6.0%	5.9%	3.7%	3.5%
		Desktop	22.0%	13.2%	6.5%	19.0%	12.9%	8.3%
		Console	18.2%	9.3%	6.4%	10.7%	2.6%	1.6%
	TGM 720p	WebBrowing	26.0%	23.2%	13.3%	9.3%	7.9%	5.5%
		Map	11.4%	8.8%	5.5%	24.6%	9.7%	6.7%
		Programming	7.1%	4.4%	1.5%	1.8%	0.2%	0.1%
		SlideShow	3.0%	2.9%	3.4%	0.4%	0.3%	0.3%
	MC 1440p	BasketballScreen	4.3%	2.4%	1.2%	1.3%	0.1%	0.0%
		MissionControlClip2	1.7%	1.4%	1.4%	0.9%	0.2%	0.0%
	MC 1080p	MissionControlClip3	5.0%	3.9%	1.9%	1.0%	0.1%	0.0%
	AN 1080p	Robot	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	CC 1080p	EBURainFruits	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		Kimono1	0.0%	-0.1%	-0.1%	0.0%	0.0%	0.0%
YCbCr	TGM 1080p	FlyingGraphics	12.9%	6.2%	5.0%	4.6%	2.9%	2.5%
		Desktop	23.4%	14.9%	7.2%	23.8%	16.4%	10.8%
		Console	25.8%	13.4%	8.2%	16.1%	6.1%	4.6%
	TGM 720p	WebBrowing	21.6%	20.2%	13.1%	13.3%	11.3%	7.8%
		Map	8.7%	6.7%	3.9%	21.7%	8.8%	5.9%
		Programming	8.3%	5.2%	2.2%	1.2%	0.1%	0.0%
		SlideShow	4.9%	4.1%	3.6%	0.3%	0.2%	0.2%
	MC 1440p	BasketballScreen	7.0%	4.8%	2.8%	1.4%	0.1%	0.0%
		MissionControlClip2	2.7%	2.4%	1.9%	0.4%	0.1%	0.0%
	MC 1080p	MissionControlClip3	6.7%	5.2%	2.8%	0.8%	0.1%	0.0%
	AN 1080p	Robot	-0.1%	-0.2%	0.0%	0.0%	0.0%	0.0%
	CC 1080p	EBURainFruits	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
		Kimono1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Overall			9.0%	6.1%	3.8%	6.1%	3.2%	2.2%

References

[1] *ITU-T H.265, ISO IEC MPEG-H Part2: high efficiency video coding*

[2] W. Pu, X. Guo, P. Onno, P. Lai, J. Xu, *Suggested software for the AHG on investigation of palette mode coding tools,* JCTVC-P303, San Jose, USA, Jan. 2014

[3] ISO/IEC JTC1/SC29/WG11 and ITU-T Q6/SG16, MPEG2014/N14175/VCEG-AW90, *Joint Call for Proposals for Coding of Screen Content*, San Jose, USA, Jan. 2014

[4] C. Pang, J. Sole, and L. Guo, *Non-RCE3: Intra motion compensation with 2-D MVs*. JCTVC-N0256, Vienna, Austria, Aug. 2013

[5] L. Zhang, J. Chen, J. Sole, M. Karczewicz, X. Xiu, Y. He, and Y. Ye, *SCCE5 Test 3.2.1: In-loop color-space transform*. JCTVC-R0147, Sapporo, Japan, July 2014

[6] P. Onno, X. Xiu, Y.-W. Huang, and R. Joshi, *Suggested combined software and text for run-based palette mode*, JCTVC-R0348, Sapporo, Japan, July 2014

[7] R. Joshi, J. Xu, *Screen content coding draft text 1*. JCTVC-R1005, Sapporo, Japan, July 2014

[8] L. Guo, W. Pu, J. Sole, M. Karczewicz, and R. Joshi, *Color palette for screen content coding*, in Proc. IEEE ICIP'14, Paris, France, Oct. 2014.

[9] R. Joshi, J. Xu, R. Cohen, S. Liu, Z. Ma, and Y. Ye, *Screen content coding test model 2 encoder description (SCM 2)*, JCTVC-R1014, Sapporo, Japan, July 2014

[10] W. Pu, F. Zou, R. Joshi, M. Karczewicz, and *J. Sole, AHG10: Simplification of Palette Based Coding*, JCTVC-Q0047, Valencia, Spain, April 2014

[11] C. Gisquet, G. Laroche, T. Poirier, and P. Onno, *SCCE3 Test C.2: Combination of palette coding tools*, JCTVC-R0086, Sapporo, Japan, July 2014

[12] M. Karczewicz, W. Pu, V. Seregin, R. Joshi, and J. Sole, *SCCE3 Test A.8: Improvements on palette prediction vector signaling*, JCTVC-R0063, Sapporo, Japan, July 2014

[13] Y. He, X. Xiu, Y. Ye, and C.-M. Tsai, *SCCE3 Test A.6: Palette table prediction*. JCTVC-R0166, Sapporo, Japan, July 2014

[14] P. Lai, S. Liu, T.-D. Chuang, Y.-W. Huang, and S. Lei, *Non-RCE4: Major color table (palette) sharing*, JCTVC-P0153, San Jose, USA, Jan. 2014

[15] C. Gisquet, G. Laroche, and P. Onno, *AhG10: Palette predictor stuffing*, JCTVC-Q0063, Valencia, Spain, Apr. 2014

[16] W. Pu, V. Seregin, R. Joshi, M. Karczewicz, and J. Sole, *SCCE3 Test B.12: Binarization of escape sample and palette index*, JCTVC-R0065, Sapporo, Japan, July 2014

[17] C. Gisquet, G. Laroche, and P. Onno, *AhG10: Palette Index Coding*, JCTVC-Q0064, Valencia, Spain, Apr. 2014

[18] C.-M. Tsai, Y. He, X. Xiu, and Y. Ye, *SCCE3 Test B.10: Palette index value mapping*. JCTVC-R0169, Sapporo, Japan, July 2014

[19] W. Pu, X. Guo, P. Onno, P. Lai, and J. Xu, *AHG 10: Suggested software for palette coding based on RExt 6.0*, JCTVC-Q0094, Valencia, Apr. 2014

[20] C.-M. Tsai, Y. He, X. Xiu, and Y. Ye, *SCCE3 Test B.9: BWT-based index grouping*, JCTVC-R0168, Sapporo, Japan, July 2014

[21] G. Bjontegaard, *Calculation of average PSNR differences between RD-curves*, VCEG-M33, Mar. 2001

[22] H. Yu, R. Cohen, K. Rapaka, and J. Xu, *Common test conditions for screen content coding*, JCTVC-R1015, Sapporo, Japan, July 2014

[23] P. Lai, P. Onno, R. Cohen, V. Seregin, X. Xiu and Z. Ma, Description of Core Experiment 1 (CE1) : Improvements of palette mode, JCTVC-S1101, Strasbourg, France, Oct. 2014