|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  21st Meeting: Warsaw, PL, 19–26 June 2015 | Document: JCTVC-U1014 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Screen content coding test model 5 (SCM 5)** | | |
| *Status:* | Output Document Approved by JCT-VC | | |
| *Purpose:* | Report | | |
| *Author(s) or Contact(s):* | Rajan Joshi Jizheng Xu Robert Cohen Shan Liu Yan Ye | Email: | [rajanj@qti.qualcomm.com](mailto:rajanj@qti.qualcomm.com) [jzxu@microsoft.com](mailto:jzxu@microsoft.com) [cohen@merl.com](mailto:cohen@merl.com) [shan.liu@mediatek.com](mailto:shan.liu@mediatek.com) [yan.ye@interdigital.com](mailto:yan.ye@interdigital.com) |
| *Source:* | Editors | | |

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

# Abstract

The JCT-VC established HEVC screen content coding test model 5 at its 21st meeting (June 19th to June 26th, 2015, Warsaw, Poland). This document serves as a source of general tutorial information and also provides an encoder-side description of test model 5.

# Introduction

The JCT-VC established HEVC screen content coding test model 5 (SCM 5) at its 21st meeting (June 19th to June 26th, 2015, Warsaw, Poland). Some of the changes from SCM 4 to SCM 5 are restriction on use of 8×8 bipred with IBC, allow non-integer IBC chroma displacement vectors, allowing the current picture to appear in both lists, increasing the DPB size to accommodate unfiltered current picture, control adaptive colour transform at the TU level rather than the CU level, and creation of SCC profiles. This document describes only the additional encoder aspects that were adopted during HEVC screen content coding (SCC) extensions development. For an understanding of the encoder upon which SCM 5 is built, please refer to the HEVC test model 16 update 3 of encoder description, JCTVC-U1002 [1], and the reference software, HEVC-16.6+SCM-5.2, which can be accessed via

<https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.6+SCM-5.2>

# Scope

This document provides an encoder-side description of the HEVC Screen content coding test model 5, which serves as a tutorial for the encoding model implemented in the HM-16.6+SCM-5.2 software. The purpose of this document is to share a common understanding of the reference encoding methods supported in the HM-16.6+SCM-5.2 software, in order to facilitate the assessment of the technical impact of the proposed new technologies during the standardization process. The common test conditions and software reference configurations that should be used for experimental work are described in JCTVC‑U1015 [2].

# Test model description

## General overview

The only changes in SCM 1 with respect to the HEVC Range extensions test model 7 [3] are in block vector search for IBC and inter modes. The search method for the inter mode is modified to make it more suitable for screen content. For the IBC mode, the chroma SAD is also used in the search for the optimal block vector (BV). Additionally the search area for the block vectors for the IBC and inter modes is expanded to include the entire picture.

The main changes from SCM 1 to SCM 2 are the addition of palette mode, residual adaptive colour transform, modifications to the block vector predictor, and block vector difference coding for IBC.

The main new coding tool from SCM 2 to SCM 3 is adaptive motion vector resolution. Other aspects of the design, such as the hash-based inter search and IBC search, palette coding, and adaptive colour transform, have also been improved.

The main changes from SCM 3 to SCM 4 are unification of intra block copy (IBC) and inter signalling, grouping of bypass bins for palette indices and escape values, and enabling the palette mode for non-4:4:4 chroma formats. Many other aspects such as palette derivation, palette predictor initialization, have also been improved.

The main changes from SCM 4 to SCM 5 are restriction on use of 8×8 bipred with IBC, allow non-integer IBC chroma displacement vectors, allowing the current picture to appear in both lists, increasing the DPB size to accommodate unfiltered current picture, control adaptive colour transform at the TU level rather than the CU level, and creation of SCC profiles. There are other small changes to the palette mode and IBC mode as well as new VUI and SEI messages.

## IBC block vector search

In order to evaluate the rate-distortion (RD) cost of using the IBC mode, for each CU, block matching (BM) is performed at the encoder to find the optimal block vector. In SCM, first a local area search is performed. This is followed by a search over the entire picture for certain CU sizes.

### Local block vector search for IBC mode

Compared to the HEVC Range extensions test model 7, the following modifications are made to the local block vector search in SCM. In order to find the optimal block vector from the local region, luma as well as chroma information is utilized. In the first step, the four best block vectors are selected according to their RD cost, where

.

Note that only the SAD of the luma component is used in this step. In the second step, both the luma and chroma components are used in the calculation of the SAD for the four best block vectors selected from step 1. The block vector with the minimum RD cost is selected as the locally optimal block vector, . The RD cost in this step is calculated as

.

The RD cost corresponding to  is denoted by.

### Global block vector search for IBC mode

In addition to the local search, global block vector search is performed for 8×8 and 16×16 blocks. The global search area is a portion of the reconstructed current picture before loop filtering, as depicted in Figure 1. Additionally, when slices/tiles are used, the search area is further restricted to be within the current slice/tile. For 16×16 blocks, only a one-dimensional search is conducted over the entire picture. This means that only the block vectors with one zero component are searched, i.e. the search is horizontal or vertical only. For 8×8 blocks, a hash-based search is used to speed up the full picture search. The bit-length of the hash table entry is 16. Each node in the hash table records the position of each block vector candidate in the picture. With the hash table, only the block vector candidates having the same hash entry value as that of the current block are examined.



Figure 1: IBC prediction area

The 16-bit hash entries for the current block and the reference block are calculated using the original pixel values. Let Grad denote the gradient of an 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

,

where MSB(X, n) represents the *n* most significant bits of X.

For 8×8 and 16×16 blocks, let the block vector with the minimum RD cost corresponding to the full-picture search be denoted by  and the corresponding RD cost be denoted by. Then,  and  are compared to choose the block vector with the minimum RD cost.

### Fast block vector search for IBC mode

In addition to the local and global block vector search, some fast search and early termination methods are employed. The fast IBC search is performed after evaluating the RD cost of inter mode, if the residual of inter prediction is not zero.

In the fast search, the SAD-based RD costs of using a set of block vector predictors are calculated. The set includes the five spatial neighbouring block vectors as utilized in inter merge mode (as shown in Figure 2 (a)) and the last two coded block vectors. In addition, the derived block vectors of the blocks pointed to by each of the aforementioned block vector predictors are also included (see Figure 2 (b)). This fast search is performed before the evaluation of intra prediction mode. It is applied only to 2Nx2N partition of various CU sizes.

 

1. Spatial BV predictors (b) Derived BV predictors

Figure : IBC prediction area

If the residue of fast IBC search is not zero, then regular intra prediction mode will be evaluated followed by a full range IBC search as described in sections 3.2.1 and 3.2.2.

### IBC block vector signalling

In SCM 4, the block vector signalling for the IBC mode is unified with the inter signalling. This is accomplished by adding the current picture to the reference picture list. Before decoding the current slice, the current picture is marked as a long-term reference picture. Then, it is converted back to a short-term reference picture after the decoding of the current picture is complete. The signalling and coding methods, including merge/AMVP signalling, AMVP derivation and MVD coding, are the same as in the inter case, with the difference that the motion vectors for IBC mode are required to be integer motion vectors. In case of 4:2:0 and 4:2:2 chroma formats, the chroma block vector is obtained by rounding to the nearest integer. An SPS flag, curr\_pic\_as\_ref\_enabled\_flag, indicates whether the current picture can be used as a reference to code itself.

In SCM 5, the following IBC aspects were changed:

* A disabling flag is added for IBC at the picture level.
* When the current picture is in one of the reference picture lists, it is necessary to additionally store the unfiltered version of the current picture. When a block uses bi-prediction, this can cause the worst case memory bandwidth to exceed that of HEVC v1. To prevent this, 8×8 bi-prediction is prohibited when the value of use\_integer\_mv\_flag is equal to 0.
* Non-integer IBC chroma displacement vectors are allowed.
* The current picture may appear both in list 0 and list 1, however weighted prediction is disabled when one of the motion vectors points to the current picture.
* In SCM 5, I-slices are possible even when IBS is enabled at the picture level.
* The current picture is placed at the last position in the reference picture list and the list is long enough to contain it.
* When in-loop filtering is disabled for the current picture and IBC is enabled, the unfiltered current picture is considered a part of DPB. This is in addition to the space used by the current picture. The maxDpbPicBuf is increased by 1 (from 6 to 7). This preserves the maximum depth of the referencing hierarchy.

o

o Store MVs in quarter-pel resolution always (i.e., for ordinary MVs, integer-only MVs, and IBC MVs) and derive integer-valued MVs by first computing a quarter-pel MV prediction, then right shifting it (with a rounding offset) to convert it to integer precision, adding integer-valued MV delta, then left-shifting it so that it is scaled for storage as a value with quarter-pel-resolution, and remove clipping of the value of MVs (U0081 method #1).

.

## Inter block search

Compared to the HEVC Range extensions test model 7, SCM modifies the inter block search in two ways. The inter search is modified to adapt to the characteristics commonly found in screen content sequences. Furthermore, the inter block search is extended to the whole picture using hash-based techniques.

### Inter search modification

The inter search in HEVC Range extensions test model 7 is modified in the following three ways to better match with the characteristics of the screen content sequences.

* Multistage approximate SAD computation: To begin with, the SAD between the current block and the candidate block is computed using only 2 lines. This SAD value is normalized to account for the subsampling and the normalized SAD is used to calculate the RD cost. If the RD cost is greater than or equal to the best RD cost so far, the search moves on to the next candidate block. Otherwise, another SAD is calculated by taking two additional lines into account and the whole process is repeated. Thus, for a 16×16 block, SAD is computed based on 2 lines (0 and 8), 4 lines (0, 4, 8, and 12), 8 lines and 16 lines of the block.
* Modified initial search: Due to the non-monotonic error pattern exhibited by non-camera-captured sequences, the exponentially expanding diamond that is used in the HEVC Range extensions test model 7 is not efficient as it may either miss one of the best candidates or falsely infer the best candidate to be within the initial search region around the best predictor. Instead, in SCM 1, a uniform diamond pattern search is employed in a smaller search area around the best predictor to better capture the local minima.

In HEVC Range extensions test model 7, an initial block vector is derived from available predictors based on the minimum RD cost. Then, exponentially expanding diamonds up to size 64 are tested as candidates. Instead, in SCM 1, the search space is restricted to  around the best predictor position. Then, this restricted search space is uniformly traversed with a step-size of 4 and fixed diamonds as shown in Figure 4.

* Modified early skip detection: In HEVC Range extensions test model 7 fast inter search, the early skip is activated when the optimal motion vector for 2*N*×2*N* mode is (0, 0) or when the residual signal of merge motion vector is less than a threshold. In SCM 1, early skip detection is based only on the residual signal of the merge motion vector.



Figure : Search candidates for the modified method

### Hash-based inter search

Hash-based search is applied only to 2N×2N blocks. An eighteen bit hash based on original pixels is used. The first 2 bits are determined by the block size, e.g. 00 for 8x8, 01 for 16x16, 10 for 32x32, and 11 for 64x64. The remaining 16 bits are determined by the original pixels.

For one block, two hash values are calculated in a similar way but with different CRC truncated polynomials. The first hash value is used for retrieval and the second hash value is used to exclude some of the hash conflicts. The hash value is calculated as follows:

* For each row, calculate the 16-bit CRC value for all the pixels Hash[i].
* Group the row hash values together (Hash[0]Hash[1]…) and then calculate the 24-bit CRC value H.
* The lower 16 bits of H will be used as the lower 16 bits of hash value of the current block.

To avoid the scenario when one hash value corresponds to many entries, the non-aligned blocks, i.e. blocks with start position (x, y) and block size *w*x*h* satisfying (x%w != 0 || y%h != 0), satisfying one of the following conditions are not added into the hash table:

* There is only one pixel value in every row; or
* There is only one pixel value in every column.

After building the hash table, the motion search is performed as follows. For each 2N×2N block,

* Perform hash based search first.
* If hash match is found, skip the normal integer pixel motion search. Otherwise, perform normal integer pixel motion search.

Early termination based on hash search is also applied. If all of the following conditions are satisfied, the RD optimization process will be terminated without checking other modes and CU splitting.

* Hash match is found.
* The quality of the reference block is no worse than the expected quality of the current block (the QP of the reference block is no greater than the QP of the current block).
* Current CU depth is 0.

## Palette mode

The palette mode was adopted into the HEVC SCC test model 2 at the 18th JCT-VC meeting (June 30th to July 9th, 2014, Sapporo, Japan). The palette mode is signalled at the CU level and is typically used when most of the pixels in the CU can be represented by a small set of representative colour values.

A brief overview of the palette mode for lossy and lossless coding, as well as some encoder side considerations will be given next. More details on the palette mode can be found in HEVC Screen Content Coding Draft Text 3 [4] and the contribution JCTVC-R0348.

### Overview of palette mode

The basic idea behind a palette mode is that the samples in the CU are represented by a small set of representative colour values. This set is referred to as the palette. It is also possible to indicate a sample that is outside the palette by signalling an escape symbol followed by (possibly quantized) component values. This is illustrated in Figure 5.



Figure : Example of a block coded in palette mode

In this example, the palette size is 4. The first 3 samples use palette entries 2, 0, and 3, respectively, for reconstruction. The blue sample represents an escape symbol. A CU level flag, palette\_escape\_val\_present\_flag, indicates whether any escape symbols are present in the CU. If escape symbols are present, the palette is augmented by one and the last index is assigned to the escape symbol. Thus, in Figure 5, index 4 is assigned to the escape symbol.

For decoding a palette-coded block, the decoder needs to have the following information:

* Palette entries
* Palette indices.

If a palette index corresponds to the escape symbol, additional component escape values (possibly quantized) are signalled.

In addition, on the encoder side, it is necessary to derive the appropriate palette to be used with that CU.

### Palette derivation

In the SCM-3.0 software, for the derivation of the palette for lossy coding, a modified k-means clustering algorithm is used. The first sample of the block is added to the palette. Then, for each subsequent sample from the block, the sum of absolute distances (SAD) from each of the current palette entries is calculated. If the distortion for each of the components is less than a threshold value for the palette entry corresponding to the minimum SAD, the sample is added to the cluster belonging to the palette entry. Otherwise, the sample is added as a new palette entry. When the number of samples mapped to a cluster exceeds a threshold, a centroid for that cluster is calculated and becomes the palette entry corresponding to that cluster.

In the next step, the clusters are sorted in a decreasing order of frequency. Then, the palette entry corresponding to each entry is updated. Normally, the cluster centroid is used as the palette entry. But a rate-distortion analysis is performed to analyze whether any entry from the palette predictor (described in section 3.4.3) may be more suitable to be used as the updated palette entry instead of the centroid when the cost of coding the palette entries is taken into account. This process is continued till all the clusters are processed or the maximum palette size is reached. Finally, if a cluster has only a single sample and the corresponding palette entry is not in the palette predictor, the sample is converted to an escape symbol. Additionally, duplicate palette entries are removed and their clusters are merged.

For lossless coding, a different derivation process is used. A histogram of the samples in the CU is calculated. The histogram is sorted in a decreasing order of frequency. Then, starting with the most frequent histogram entry, each entry is added to the palette. Histogram entries that occur only once are converted to escape symbols if they are not a part of the palette predictor.

After palette derivation, each sample in the block is assigned the index of the nearest (in SAD) palette entry. Then, the samples are assigned to 'INDEX' or 'COPY\_ABOVE' mode (described in section 3.4.4). For each sample for which either 'INDEX' or 'COPY\_ABOVE' mode is possible, the run for each mode is determined. Then, the cost (in terms of average bits per sample position) of coding the mode, the run and possibly the index value (for 'INDEX' mode) is calculated. The mode for which the cost is lower is selected. The decision is greedy in the sense that future runs and their costs are not taken into account.

### Coding of the palette entries

For coding of the palette entries, a palette predictor is maintained. The maximum size of the palette as well as the palette predictor is signalled in the SPS. In SCM 4, a palette\_predictor\_initializer\_present\_flag is introduced in the PPS. When this flag is 1, entries for initializing the palette predictor are signalled in the bitstream. The palette predictor is initialized at the beginning of each CTU row, each slice and each tile. Depending on the value of the palette\_predictor\_initializer\_present\_flag, the palette predictor is reset to 0 or initialized using the palette predictor intializer entries signalled in the PPS.

For each entry in the palette predictor, a reuse flag is signalled to indicate whether it is part of the current palette. This is illustrated in Figure 6. The reuse flags are sent using run-length coding of zeros. After this, the number of new palette entries are signalled using exponential Golomb code of order 0. Finally, the component values for the new palette entries are signalled.



Figure : Use of palette predictor to signal palette entries

### Coding of palette indices

The palette indices are coded using horizontal and vertical traverse scans as shown in Figure 7. The scan order is explicitly signalled in the bitstream using the palette\_transpose\_flag. For the rest of the subsection it is assumed that the scan is horizontal.



Figure : Horizontal and vertical traverse scans

The palette indices are coded using two main palette sample modes: 'INDEX' and 'COPY\_ABOVE'. As explained previously, the escape symbol is also signalled as an 'INDEX' mode and assigned an index equal to the maximum palette size. The mode is signalled using a flag except for the top row or when the previous mode was 'COPY\_ABOVE'. In the 'COPY\_ABOVE' mode, the palette index of the sample in the row above is copied. In the 'INDEX' mode, the palette index is explicitly signalled. For both 'INDEX' and 'COPY\_ABOVE' modes, a run value is signalled which specifies the number of *subsequent* samples that are also coded using the same mode. When escape symbol is part of the run in 'INDEX' or 'COPY\_ABOVE' mode, the escape component values are signalled for each escape symbol. The coding of palette indices is illustrated in Figure 8.

Prior to SCM 4, the signalling order was: palette sample mode (if necessary), index value (if necessary), run, and finally, escape component values for the escape symbols in the run. This order is modified in SCM 4 in order to group the bypass bins for indices and escape samples. This is accomplished as follows. First the number of index values for the CU is signalled. This is followed by signalling of the actual index values for the entire CU using truncated binary coding. Both the number of indices as well as the the index values are coded in bypass mode. This groups the index-related bypass bins together. Then the palette sample mode (if necessary) and run are signalled in an interleaved manner. Finally, the component escape values corresponding to the escape samples for the entire CU are grouped together and coded in bypass mode.

An additional syntax element, last\_run\_type\_flag, is signalled after signalling the index values. This syntax element, in conjuction with the number of indices, eliminates the need to signal the run value corresponding to the last run in the block.

In SCM 4, the palette mode is also enabled for 4:2:2, 4:2:0, and monochrome chroma formats. The signalling of the palette entries and palette indices is almost identical for all the chroma formats. In case of non-monochrome formats, each palette entry consists of 3 components. For the monochrome format, each palette entry consists of a single component. For subsampled chroma directions, the chroma samples are associated with luma sample indices that are divisible by 2. After reconstructing the palette indices for the CU, if a sample has only a single component associated with it, only the first component of the palette entry is used. The only difference in signalling is for the escape component values. For each escape sample, the number of escape component values signalled may be different depending on the number of components associated with that sample.



Figure : Coding of palette indices

## Adaptive Colour Transform

The Adaptive Colour Transform (ACT) was adopted into the HEVC SCC test model 2 at the 18th JCT-VC meeting (June 30th to July 9th, 2014, Sapporo, Japan). ACT performs in-loop colour space conversion in the prediction residual domain using colour transform matrices based on the YCoCg and YCoCg-R colour spaces. ACT is turned on or off adaptively at the CU level using the flag cu\_residual\_act\_flag. ACT can be combined with Cross Component Prediction (CCP), which is another inter component de-correlation method already supported in HEVC RExt [3]. When both are enabled, ACT is performed after CCP at the decoder, as shown in Figure 9.

A brief overview of ACT, as well as some encoder side considerations will be given next. More details on ACT can be found in SCC Draft 4 [4] and the contribution JCTVC-R0147. In SCM 5, the signalling of ACT is moved from the CU level to the TU level to align it with CCP.



Figure : SCC decoder flow of in-loop ACT

### Colour space conversion in ACT

The colour space conversion in ACT is based on the YCoCg-R transform. Both lossy coding and lossless coding (cu\_transquant\_bypass\_flag = 0 or 1) use the same inverse transform, but an additional 1-bit left shift is applied to the Co and Cg components in the case of lossy coding. Specifically, the following colour space transforms are used for forward and backward conversion for lossy and lossless coding:

Forward transform for lossy coding (non-normative):



Forward transform for lossless coding (non-normative):



Backward transform (normative):



The forward colour transform is not normalized, with its norm being roughly equal to  for Y and Cg and equal to for Co. In order to compensate for the non-normalized nature of the forward transform, delta QPs of (−5, −3, −5) are applied to (Y, Co, Cg) , respectively. In other words, for a given “normal” QP for the CU, if ACT is turned on, then the quantization parameter is set equal to (QP − 5, QP − 3, QP −5) for (Y, Co, Cg), respectively. The adjusted quantization parameter only affects the quantization and inverse quantization of the residuals in the CU. For deblocking, the “normal” QP value is still used. Clipping to 0 is applied to the adjusted QP values to ensure that they will not become negative. Note that this QP adjustment is only applicable to lossy coding, as quantization is not performed in lossless coding (cu\_transquant\_bypass\_flag=1). In SCM 4, PPS/slice-level signalling of additional QP offset values is introduced. These QP offset values may be used instead of (−5, −3, −5) for CUs when adaptive colour transform is applied.

When the input bit-depths of the colour components are different, appropriate left shifts are applied to align the sample bit-depths to the maximal bit-depth during ACT, and appropriate right shifts are applied to restore the original sample bit-depths after ACT.

### Encoder optimization for ACT

Care is taken on the encoder side when performing ACT, in order to avoid doubling the encoder complexity by searching over all the possible modes twice - in both the original colour space and the converted colour space. The following fast encoding methods are employed in SCM 3.

* For intra mode coding, the best luma and chroma modes are decided once and shared between the two colour spaces. Further, when ACT is on (cu\_residual\_act\_flag = 1), chroma components will share the same prediction mode as the luma component. That is, the only chroma prediction mode tested is the DM mode; other chroma prediction modes are not checked. This is the same as with the CCP encoder in HEVC Rext.
* For IBC and inter modes, block vector search or motion estimation is performed only once. The block vectors and motion vectors are shared between the two colour spaces. Further, for RGB sequences, block vector search for IBC mode is conducted in the converted (i.e., YCoCg) colour space.
* The order of checking the rate-distortion cost of enabling and disabling ACT depends on the original video content. For RGB sequences, the RD cost for enabling ACT is checked first; for YCbCr sequences, the RD cost for disabling ACT is checked first. Rate distortion cost of the second colour space is invoked only when there is at least one non-zero coefficient in the first colour space.
* RD cost of parent CU is used to decide whether to check the RD cost in the second colour space for the current CU. For example, if for the parent CU, the RD cost of the first colour space is smaller than that of the second colour space, then for the current CU, the second colour space is not considered.

Another encoder optimization technique for ACT is the chroma lambda adjustment method. Specifically, the chroma lambda used to calculate RD cost is increased compared to that for the luma component. The chroma lambda value is modified based on the input QP, using the following equation:



where and Table 1 specifies the mapping between QP and delta(QP).

Table 1: Specification of delta(QP) used in chroma lambda adjustment for ACT

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| QP | [0, 14] | [15, 29] | [30, 36] | [37, 38] | [39, 40] | [41, 42] | [43, 52] |
| delta(QP) | 0 | -1 | -2 | -3 | -4 | -5 | -6 |

## Adaptive motion vector resolution decision

Adaptive MV resolution allows the MVs of an entire picture to be signalled in either quarter-pel precision (same as HEVC version 1) or integer-pel precision. Hash based motion statistics are kept and checked in order to properly decide the appropriate MV resolution for the current picture without relying on multi-pass encoding. In SCM 4, a new syntax element, motion\_vector\_resolution\_control\_idc, is introduced. If motion\_vector\_resolution\_control\_idc is equal to 0/1, the motion vector precision for the entire coded video sequence is quarter-pel/integer-pel. When motion\_vector\_resolution\_control\_idc is equal to 2, the motion vector precision is signalled in the slice header.

To decide the MV precision of one picture, the encoder performs the following check with the help of hashes. For every non-overlapped 8×8 block in a picture, the encoder checks whether it can find a matching block by hash in the first reference picture in list 0. The blocks are classified into the following categories:

* *C*: number of blocks matching with collocated block.
* *S*: number of blocks not matching with collocated block but belong to smooth region. For smooth region, it means every column has a single pixel value or every row has a single pixel value.
* *M*: number of blocks not belonging to C or S but can find a matching block by hash value.

*T* is the total number of blocks in one picture. *CSMRate* = (*C*+*S*+*M*)/*T*, *MRate* = *M*/*T*. *AverageCSMRate* is the average *CSMRate* of current picture and the previous 31 pictures. *AverageMRate* is the average *MRage* of the current picture and the previous 31 pictures.

The MV resolution is determined as:

* If *CSMRate* < 0.8, use quarter-pel MV.
* Otherwise, if *C* == *T*, use integer-pel MV.
* Otherwise, if *AverageCSMRate* < 0.95, use quarter-pel MV.
* Otherwise, if *M* > (*T*−*C*−*S*)/3, use integer-pel MV.
* Otherwise, if *CSMRate* > 0.99 and *MRate* > 0.01, use integer-pel MV.
* Otherwise, if *AverageCSMRate* + *AverageMRate* > 1.01, use integer-pel MV.

Otherwise, use quarter-pel MV.

In SCM 5, one other change that applies to adaptive motion vector resolution as well as IBC is as follows. Motion vector processing and storage is unified for IBC and inter. The motion vectors (MVs) are always stored in quarter-pel precision for ordinary MVs, integer-only MVs and IBC MVs. The integer-valued MVs are derived by first computing a quarter-pel MV prediction, then right shifting it (with a rounding offset) to convert it to integer precision, adding integer-valued MV difference, then left-shifting it so that it is scaled for storage as a value with quarter-pel precision. Also, MV clipping used when use\_integer\_mv\_flag is equal to 1, to guarantee a valid MV range, is removed.

# References

1. JCTVC-U1002, C. Rosewarne, B. Bross, M. Naccari, K. Sharman, G. J. Sullivan (editors), "High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Update 3 of Encoder Description," Warsaw, Poland, June 2015.
2. JCTVC-U1015, H. Yu, R. Cohen, K. Rapaka, J. Xu, "Common Test Conditions for Screen Content Coding," Warsaw, Poland, June 2015.
3. JCTVC-Q1013, C. Rosewarne, K. Sharman, M. Naccari, G. J. Sullivan, "HEVC Range Extensions Test Model 7 Encoder Description," JCTVC-Q1015, Valencia, Spain, April 2014.
4. JCTVC-U1005-v2, R. Joshi, S. Liu, J. Xu, Y. Ye (editors), "HEVC Screen Content Coding Draft Text 4," Warsaw, Poland, June 2015.