|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  18th Meeting: Sapporo, JP, 30 June – 9 July 2014 | Document: JCTVC-R0217r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **SEI comments** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Information | | |
| *Author(s) or Contact(s):* | Chad Fogg 530 Lytton Ave, Suite 300 Palo Alto, CA 94301 USA | Email: | chadfogg@gmail.com |
| *Source:* | MovieLabs | | |

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

# Abstract

Editorial comments and suggestions for the chroma resampling hint SEI are provided in this information document. From the manner in which the semantics are described, the current definition of the SEI implies that it is intended for field sequence pictures only. The intent of the SEI may be instead to describe the chroma resampling filtering of the output fields that includes frame pictures with field repeats as per pic\_struct semantics in pic\_timing() SEI. The main suggestion is to clarify that the metadata provided by this SEI refer to the output fields rather than give the impression the metadata maps only to coded pictures (access units) in the bitstream.

# Introduction

The following suggestions are made with respect to the current draft of the SHVC draft [1]

Revision of this document (R0217r1) reassesses comments in light of editorial edition JCTVC-Q1008\_v5\_k11\_gb6\_jb1.docx. The sections for chroma resampling hint SEI are copied in section 3 of this document.

## ver\_field\_processing\_flag

It is requested to update the SHVC draft semantics for this element to accommodate Progressive Segmented Frames (PsF) in field sequences, such as when pic\_struct is equal to 9,10,11, or 12.

## Precision requirements

An editorial note: suggestion that the precision of the filtering operations is sufficient to meet the bit length of the se(u) filter tap elements.

# References

[1] JCTVC-Q1008 “Scalable extensions” , June 2014

[2] Society of Motion Pictures and Television Engineers, “4:2:2 / 4:2:0 Format Conversion Minimizing Color Difference Signal Degradation in Concatenated Operations — Filtering” [RP 2050-1:2012](http://standards.smpte.org/content/978-1-61482-653-8/rp-2050-1-2012/SEC1)

# Text from SHVC draft [1]

## Syntax

D.2.26 Chroma resampling filter hint SEI message syntax

|  |  |
| --- | --- |
| chroma\_resampling\_filter\_hint( payloadSize ) { | Descriptor |
| **ver\_chroma\_filter\_idc** | u(8) |
| **hor\_chroma\_filter\_idc** | u(8) |
| **ver\_filtering\_field\_processing\_flag** | u(1) |
| if( ver\_chroma\_filter\_idc = = 1 | | hor\_chroma\_filter\_idc = = 1 ) { |  |
| **target\_\_format\_idc** | ue(v) |
| if( ver\_chroma\_filter\_idc = = 1 ) { |  |
| **num\_vertical\_filters** | ue(v) |
| for( i = 0; i < num\_vertical\_filters; i++) { |  |
| **ver\_tap\_length\_minus1**[ i ] | ue(v) |
| for( j = 0; j <= ver\_tap\_length\_minus1[ i ]; j++) |  |
| **ver\_filter\_coeff**[ i ][ j ] | se(v) |
| } |  |
| } |  |
| if( hor\_chroma\_filter\_idc = = 1 ) { |  |
| **num\_horizontal\_filters** | ue(v) |
| for( i=0; i < num\_horizontal\_filters; i++) { |  |
| **hor\_tap\_length\_minus1**[ i ] | ue(v) |
| for( j=0; j <= hor\_tap\_length\_minus1[ i ]; j++) |  |
| **hor\_filter\_coeff**[ i ][ j ] | se(v) |
| } |  |
| } |  |
| } |  |
| } |  |

D.2.27 Knee function information SEI message syntax

|  |  |
| --- | --- |
| knee\_function\_info( payloadSize ) { | Descriptor |
| **knee\_function\_id** | ue(v) |
| **knee\_function\_cancel\_flag** | u(1) |
| if( !knee\_function\_cancel\_flag ) { |  |
| **knee\_function\_persistence\_flag** | u(1) |
| **mapping\_flag** | u(1) |
| **input\_d\_range** | u(32) |
| **input\_disp\_luminance** | u(32) |
| **output\_d\_range** | u(32) |
| **output\_disp\_luminance** | u(32) |
| **num\_knee\_points\_minus1** | ue(v) |
| for( i = 0; i <= num\_knee\_points\_minus1; i++ ) { |  |
| **input\_knee\_point**[ i ] | u(10) |
| **output\_knee\_point**[ i ] | u(10) |
| } |  |
| } |  |
| } |  |

## Semantics

D.3.26 Chroma resampling filter hint SEI message semantics

The chroma resampling filter hint SEI message signals one downsampling process and one upsampling process for the chroma components of decoded pictures. When the sampling processes signalled in the chroma resampling filter hint SEI message are used, for any number of upsampling and downsampling iterations performed on the decoded pictures, the degradation of the colour components is expected to be minimized.

The chroma resampling filter hint SEI message shall not be present in a CLVS that has chroma\_format\_idc equal to 0.

It is a requirement of bitstream conformance that when a chroma resampling filter hint SEI message is present in a CLVS chroma\_sample\_loc\_type\_top\_field shall be equal to chroma\_sample\_loc\_type\_bottom\_field in the same CLVS.

**ver\_chroma\_filter\_idc** identifies the vertical components of the downsampling and upsampling sets of filters as specified in Table D-2. Based on the value of ver\_chroma\_filter\_idc, the values of verFilterCoeff[ ][ ] are derived from Table D-6. The value of ver\_chroma\_filter\_idc shall be in the range of 0 to 2, inclusive. Values of ver\_chroma\_filter\_idc greater than 2 are reserved for future use by ITU-T | ISO/IEC.

When ver\_chroma\_filter\_idc is equal to 0, the chroma resampling filter in the vertical direction is unspecified.

When chroma\_format\_idc is equal to 1, ver\_chroma\_filter\_idc shall be equal to 1 or 2.

When hor\_chroma\_filter\_idc is equal to 0, ver\_chroma\_filter\_idc shall not be equal to 0.

Table D‑2 – ver\_chroma\_filter\_idc values

|  |  |
| --- | --- |
| Value | Description |
| 0 | Unspecified |
| 1 | Filters signalled by ver\_filter\_coeff[ ][ ] |
| 2 | Filters as described in SMPTE RP 2050-1:2012 |
| >2 | Reserved |

**hor\_chroma\_filter\_idc** identifies the horizontal components of the downsampling and upsampling sets of filters as specified in Table D-3. Based on the value of hor\_chroma\_filter\_idc, the values of horFilterCoeff[ ][ ] are derived from Table D-7. The value of hor\_chroma\_filter\_idc shall be in the range of 0 to 2, inclusive. Values of hor\_chroma\_filter\_idc greater than 2 are reserved for future use by ITU-T | ISO/IEC.

When hor\_chroma\_filter\_idc is equal to 0, the chroma resampling filter in the horizontal direction is unspecified.

When chroma\_format\_idc is equal to 3, hor\_chroma\_filter\_idc shall be equal to 1 or 2.

When ver\_chroma\_filter\_idc is equal to 0, hor\_chroma\_filter\_idc shall not be equal to 0.

When chroma\_format\_idc is equal to 2 and ver\_chroma\_filter\_idc is equal to 2, hor\_chroma\_filter\_idc shall be equal to 0.

Table D‑3 – hor\_chroma\_filter\_idc values

|  |  |
| --- | --- |
| Value | Description |
| 0 | Unspecified |
| 1 | Filters signalled by hor\_filter\_coeff[ ][ ] |
| 2 | Filters as described in the 5/3 filter description of ITU-T Rec. T.800 | ISO/IEC15444-1 |
| >2 | Reserved |

**ver\_filtering\_field\_processing\_flag** indicates whether the vertical operations of the downsampling and the upsampling sets of filters should be applied on a field or frame basis.

Based on the value of ver\_filtering\_field\_processing\_flag and seq\_field\_flag, the following applies:

* If seq\_field\_flag is equal to 1, each decoded field should be filtered independently.
* Otherwise (seq\_field\_flag is equal to 0), if ver\_filtering\_field\_processing\_flag is equal to 1, for each decoded frame, all the chroma components with the same parity should be filtered independently from the chroma components with a different parity (i.e. the filter shall be applied on a field basis).
* Otherwise (seq\_field\_flag is equal to 0 and ver\_filtering\_field\_processing\_flag is equal to 0), the filtering process should be applied on all chroma components without consideration of their parity (i.e. the filter should be applied on a frame basis).

NOTE1 – If ver\_filtering\_field\_processing\_flag is equal to 0, then even in the case when field\_seq\_flag is equal to 1, each output decoded field is still considered as a frame during the filtering process.

The variable chromaSampleLocType is derived as follows:

* If chroma\_format\_idc is equal to 1, chromaSampleLocType is set equal to chroma\_sample\_loc\_type\_top\_field.
* Otherwise (chroma\_format\_idc is not equal to 1), chromaSampleLocType is set equal to 0.

When chromaSampleLocType is greater than 1, ver\_chroma\_filter\_idc shall not be equal to 2.

When chromaSampleLocType is equal to 1, 3 or 5, hor\_chroma\_filter\_idc shall not be equal to 2.

**target\_format\_idc** indicates the output sampling format of the chroma components (i.e. the position of chroma components relative to that of the luma components) after filtering. The sampling format indicated by each value of target\_format\_idc is defined in Table D-4. [Ed. (JB): Awkward. I’m not sure what exactly it is trying to say.] The value of target\_format\_idc shall be in the range of 1 to 3, inclusive. The value of target\_format\_idc shall not be equal to the value of chroma\_format\_idc.

When not present, the value of target\_format\_idc is inferred as follows:

* If chroma\_format\_idc is equal to 1:
  + If ver\_chroma\_filter\_idc is equal to 2:
    - If hor\_chroma\_filter\_idc is not equal to 2, the value of target\_format\_idc is inferred to be equal to 2.
    - Otherwise (hor\_chroma\_filter\_idc is equal to 2), the value of target\_format\_idc is inferred to be equal to 3.
* Otherwise, if chroma\_format\_idc is equal to 2:
  + If ver\_chroma\_filter\_idc is equal to 2, the value of target\_format\_idc is inferred to be equal to 1.
  + Otherwise (ver\_chroma\_filter\_idc is not equal to 2), the value of target\_format\_idc is inferred to be equal to 3.
* Otherwise, if chroma\_format\_idc is equal to 3:
  + If ver\_chroma\_filter\_idc is equal to 2, the value of target\_format\_idc is inferred to be equal to 1.
  + Otherwise (ver\_chroma\_filter\_idc is not equal to 2), the value of target\_format\_idc is inferred to be equal to 2.

When chroma\_format\_idc is equal to 2 or 3 and target\_format\_idc is equal to 2 or 3, ver\_chroma\_filter\_idc shall be equal to 0.

When chroma\_format\_idc is equal to 1 or 2 and target\_format\_idc is equal to 1 or 2, hor\_chroma\_filter\_idc shall be equal to 0.

Table D-4 Chroma sampling format indicated by target\_format\_idc

|  |  |
| --- | --- |
| target\_format\_idc | Chroma sampling format |
| 1 | 4:2:0 |
| 2 | 4:2:2 |
| 3 | 4:4:4 |

NOTE2 – The logic associating sampling formats to numeric values of target\_format\_idc is the same as the one used to associate sampling formats to numeric values of chroma\_format\_idc, described in subclause 6.2.

The variable upsamplingFlag is derived as follows:

* If chroma\_format\_idc is greater than target\_format\_idc, upsamplingFlag is set equal to 0.
* Otherwise (chroma\_format\_idc is less than target\_format\_idc), upsamplingFlag is set equal to 1.

**num\_vertical\_filters** specifies the number of filters signalled for chroma downsampling and upsampling in the vertical direction. When ver\_chroma\_filter\_idc is equal to 1, the value of num\_vertical\_filters shall follow the constraints indicated in **Error! Reference source not found.**, determined by the values of chromaSampleLocType and ver\_filtering\_field\_processing\_flag.

**Table D-5 constraints on the value of num\_vertical\_filters**

|  |  |  |
| --- | --- | --- |
| Conditions | | mandatory value of num\_vertical\_filters |
| chromaSampleLocType | ver\_filtering\_field\_processing\_flag |
| 0,1 | 0 | 2 |
| 1 | 3 |
| 2,3 | 0 | 3 |
| 1 | 5 |
| 4,5 | 0 | 3 |
| 1 | 5 |

**ver\_tap\_length\_minus1**[ i ] plus 1 specifies the length of the i-th filter in the vertical direction. The value of ver\_tap\_length\_minus1[ i ] shall be in the range of 0 to 31, inclusive.

**ver\_filter\_coeff**[ i ][ j ] specifies the value of the j-th coefficient of the i-th filter in the vertical direction. The value of ver\_filter\_coeff[ i ][ j ] shall be in the range of −231 to 231 − 1, inclusive.

The variable verTapLength[ ] is derived as follows:

* If ver\_chroma\_filter\_idc is equal to 1, verTapLength[ i ] is set equal to ver\_tap\_length\_minus1[ i ] plus 1 for i in the range of 0..num\_vertical\_filters-1.
* Otherwise (ver\_chroma\_filter\_idc is equal to 2), the values of verTapLength[ ] are derived as specified in Table D‑.

The variable verFilterCoeff[ ][ ] is derived as follows:

* If ver\_chroma\_filter\_idc is equal to 1, verFilterCoeff[ i ][ j ] is set equal to ver\_filter\_coeff[ i ][ j ] for i in the range of 0..num\_vertical\_filters-1 and j in the range of 0..ver\_tap\_length\_minus1[ i ].
* Otherwise (ver\_chroma\_filter\_idc is equal to 2), the values of verFilterCoeff[ ][ ] are derived as specified in Table D‑.

**num\_horizontal\_filters** specifies the number of filters indicated for chroma downsampling and upsampling in the horizontal direction. When hor\_chroma\_filter\_idc is equal to 1, the value of num\_horizontal\_filters shall follow the constraints specified in **Error! Reference source not found.**.

**Table D-6 constraints on the value of num\_horizontal\_filters**

|  |  |
| --- | --- |
| chromaSampleLocType | mandatory value of  num\_horizontal\_filters |
| 0, 2, 4 | 3 |
| 1, 3, 5 | 2 |

**hor\_tap\_length\_minus1**[ i ] plus 1 specifies the length of the i-th filter in the horizontal direction. The value of hor\_tap\_length\_minus1[ i ] shall be in the range of 0 to 31, inclusive.

**hor\_filter\_coeff**[ i ][ j ] specifies the value of the j-th coefficient of the i-th filter in the horizontal direction. The value of hor\_filter\_coeff[ i ][ j ] shall be in the range of −231 to 231 − 1, inclusive.

The variable horTapLength[ ] is derived as follows:

* If hor\_chroma\_filter\_idc is equal to 1, horTapLength[ i ] is set equal to hor\_tap\_length\_minus1[ i ] plus 1 for i in the range of 0..num\_horizontal\_filters-1.
* Otherwise (hor\_chroma\_filter\_idc is equal to 2), the values of horTapLength[ ] are derived as specified in Table D‑.

The variable horFilterCoeff[ ][ ] is derived as follows:

* If hor\_chroma\_filter\_idc is equal to 1, horFilterCoeff[ i ][ j ] is set equal to hor\_filter\_coeff[ i ][ j ] for i in the range of 0..num\_horizontal\_filters-1 and j in the range of 0..hor\_tap\_length\_minus1[ i ].
* Otherwise (hor\_chroma\_filter\_idc is equal to 2), the values of horFilterCoeff[ ][ ] are derived as specified in Table D‑.

Table D‑7 specifies the coefficients of the chroma downsampling and upsampling sets of filters in the vertical direction when ver\_chroma\_filter\_idc is equal to 2. When ver\_chroma\_filter\_idc is equal to 2, the filter coefficient values described in SMPTE RP 2050 1:2012 are used. [Ed. (JB): Are these the SMPTE values? Still useful to explicitly mention that here, just not as a column in the table.]

Table D‑8 specifies the coefficients of the chroma downsampling and upsampling sets of filters in the horizontal direction when hor\_chroma\_filter\_idc is equal to 2. When ver\_chroma\_filter\_idc is equal to 2, the filter coefficient values described in the 5/3 filter specification part of ITU-T Rec. T.800 | ISO/IEC15444-1 are used.

Table D‑7 – Values of verFilterCoeff and verTapLength when ver\_chroma\_filter\_idc is equal to 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| chromaSampleLocType | ver\_filtering\_field\_processing\_flag | | upsamplingFlag | verFilterCoeff[ ][ ] | verTapLength[ ] |
| 0, 1 | 0 | 0 | | verFilterCoeff[ 0 ][ ] = { −3, −19, 34, 500, 500, 34, −19, −3 } | verTapLength[ 0 ] = 8 |
| 1 | | verFilterCoeff[ 1 ][ ] = { 19, 103, 1037, −135 } | verTapLength[ 1 ] = 4 |
| 1 | 0 | | verFilterCoeff[ 0 ][ ] = { −8, −26, 115, 586, 409, −48, −4, 0 } | verTapLength[ 0 ] = 8 |
| 1 | | verFilterCoeff[ 1 ][ ] = { 24, −41, 1169, −128 } | verTapLength[ 1 ] = 4 |
| verFilterCoeff[ 2 ][ ] = { −76, 783, 330, −13 } | verTapLength[ 2 ] = 4 |

[Ed. (JB): Why include the minus in the variable? It makes sense to have signaled the syntax element that way, but now that it is assigned to a variable, isn’t it less confusing to have it just be the tap length? Same for horizontal.]

Table D‑8 – Values of horFilterCoeff and horTapLength when hor\_chroma\_filter\_idc is equal to 2

|  |  |  |  |
| --- | --- | --- | --- |
| chromaSampleLocType | upsamplingFlag | horFilterCoeff[ ][ ] | horTapLength[ ] |
| 0, 2, 4 | 0 | horFilterCoeff[ 0 ][ ] = { −1, 2, 6, 2, −1 } | horTapLength[ 0 ] = 5 |
| 1 | horFilterCoeff[ 1 ][ ] = { 1 } | horTapLength[ 1 ] = 1 |
| horFilterCoeff[ 2 ][ ] = { 1, 1 } | horTapLength[ 2 ] = 2 |

The chroma resampling filtering process is modelled as follows:

* When ver\_filtering\_field\_process\_flag is equal to 1, the following applies:
* When field\_seq\_flag is equal to 0, each output frame is deinterleaved into a top field and a bottom field and each field is processed in successive order.

The variable bottomFlag is derived as follows:

* + If field\_seq\_flag is equal to 1 and pic\_struct is equal to 2, 10 or 12, bottomFlag is set equal to 1.
  + Otherwise, if field\_seq\_flag is equal to 0 and ver\_filtering\_field\_processing\_flag is equal to 1 and the output field being processed is a bottom field, bottomFlag is set equal to 1

Otherwise, bottomFlag is set equal to 0.

The variables fDv[ ][ ] and fUv[ ][ ] respectively corresponding to coefficients used for downsampling filtering and upsampling filtering in the vertical direction are derived as specified in Table D-9.

The variables fDh[ ][ ] and fUh[ ][ ] respectively corresponding to coefficients used for downsampling filtering and upsampling filtering in the horizontal direction are derived as specified in Table D-10.

* The variables phaseOffsetUp and phaseOffsetDown are derived as follows:
  + If bottomFlag is equal to 1, phaseOffsetUp is set equal to 2 and phaseOffsetDown is set equal to 1.
  + Otherwise (bottomFlag is equal to 0), phaseOffsetUp is set equal to 0 and phaseOffsetDown is set equal to 0.
* When chroma\_format\_idc is equal to 1 and target\_format\_idc is equal to either 2 or 3, the chroma upsampling filtering process in the vertical direction is applied as follows:

divUv[ 0 ] = 0  
divUv[ 1 ] = 0  
for( j = 0; j < lenUv[ 0 + phaseOffsetUp ]; j++ )  
 divUv[ 0 ] += fUv[ 0 + phaseOffsetUp ][ j ]  
for( j = 0; j < lenUv[ 1 + phaseOffsetUp ]; j++ )  
 divUv[ 1 ] += fUv[ 1 + phaseOffsetUp ][ j ]  
w0 = ( pic\_width\_in\_luma\_samples / SubWidthC ) – ( conf\_win\_right\_offset + conf\_win\_left\_offset )  
h0 = ( pic\_height\_in\_luma\_samples >> 1 ) – ( conf\_win\_top\_offset + conf\_win\_bottom\_offset )  
for( u = 0; u < w0; u++ )   
 for( v = 0; v < ( h0 << 1 ); v++ ) {  
 sum = 0   
 for( j = − ( lenUv[ v % 2 + phaseOffsetUp ] − 1 ) / 2;  
 j <= lenUv[ v % 2 + phaseOffsetUp ] / 2; j++ )  
 sum += p0[ u ][ Clip3( 0, h0 − 1, ( v  >>  1 ) + j ) ]  
 \* fUv[ v % 2 + phaseOffsetUp ][ j + ( lenUv[ v % 2 + phaseOffsetUp ] − 1 ) / 2 ]  
 p1[ u ][ v ] = ( sum + ( divUv[ v % 2 ]  >>  1 ) ) / divUv[ v % 2 ]  
 } (D‑3)

where p0[ ][ ] is the array of chroma samples in a cropped output picture before vertical chroma upsampling, and p1[ ][ ] is the array of chroma samples in a cropped output picture after vertical chroma upsampling.

When ver\_filtering\_field\_process\_flag is equal to 1 and field\_seq\_flag is equal to 0, the chroma upsampling filtering process in the vertical direction is applied to each field of the cropped output frame picture p0. Firstly p0 is deinterleaved into two fields p0Top and p0Bottom whose heights are equal to h0 >> 1. Then the chroma upsampling filtering process in the vertical direction is applied to p0Top and p0Bottom. Finally the outputs of the filtering process p1Top and p1Bottom are interleaved to form the cropped output frame picture after vertical chroma upsampling.

* When chroma\_format\_idc is equal to either 1 or 2 and target\_format\_idc is equal to 3, the chroma upsampling filtering process in the horizontal direction appliedas follows:

divUh[ 0 ] = 0  
divUh[ 1 ] = 0  
for( j = 0; j < lenUh[ 0 ]; j++ )  
 divUh[ 0 ] += fUh[ 0 ][ j ]  
for( j = 0; j < lenUh[ 1 ]; j++ )  
 divUh[ 1 ] += fUh[ 1 ][ j ]  
h0 = ( pic\_height\_in\_luma\_samples / SubHeightC ) – ( conf\_win\_top\_offset + conf\_win\_bottom\_offset )  
w0 = ( pic\_width\_in\_luma\_samples >> 1 ) – ( conf\_win\_right\_offset + conf\_win\_left\_offset )  
for( v = 0; v < h0; v++ )  
 for( u = 0; u < ( w0 << 1 ); u++ ) {  
 sum = 0 for( i = − ( lenUh[ u % 2 ] − 1 ) / 2; i  <=  lenUh[ u % 2 ] / 2; i++ )  
 sum += p0[ Clip3( 0, h0 – 1, ( u  >>  1 ) + i ) ][ v ] \* fUh[ u % 2 ][ i + ( lenUh[ u % 2 ] − 1 ) / 2 ]  
 p1[ u ][ v ] = ( sum + ( divUh[ u % 2 ] >> 1 ) ) / divUh[ u % 2 ]  
 } (D‑4)

where p0[ ][ ] is the array of chroma samples in a cropped output picture before horizontal chroma upsampling, and p1[ ][ ] is the array of chroma samples in a cropped output picture after horizontal chroma upsampling.

* When chroma\_format\_idc is equal to either 3 or 2 and target\_format\_idc is equal to 1, the chroma downsampling filtering process in the vertical direction is applied as follows:

divDv = 0  
for( j = 0; j < lenDv[ phaseOffsetDown ]; j++ )  
 divDv += fDv[ phaseOffsetDown ][ j ]  
w0 = ( pic\_width\_in\_luma\_samples / SubWidthC ) – ( conf\_win\_right\_offset + conf\_win\_left\_offset )  
h0 = pic\_height\_in\_luma\_samples – ( conf\_win\_top\_offset + conf\_win\_bottom\_offset )  
for( u = 0; u < w0; u++ )  
 for( v = 0; v < ( h0 >> 1 ); v++ ) {  
 sum = 0  
 for( j = − ( lenDv[ phaseOffsetDown ] − 1 ) / 2; j  <=  lenDv[ phaseOffsetDown ] / 2; j++ )  
 sum += p0[ u ][ Clip3( 0, h0 − 1, ( v  <<  1 ) + j ) ]  
 \* fDv[ phaseOffsetDown ][ j + ( lenDv[ phaseOffsetDown ] − 1 ) / 2 ]  
 p1[ u ][ v ] = ( sum + ( divDv >> 1 ) ) / divDv  
 } (D‑5)

where p0[ ][ ] is the array of chroma samples in a cropped output picture before vertical chroma downsampling, and p1[ ][ ] is the array of chroma samples in a cropped output picture after vertical chroma downsampling.

When ver\_filtering\_field\_process\_flag is equal to 1 and field\_seq\_flag is equal to 0, the chroma downsampling filtering process in the vertical direction is applied to each field of the cropped output frame picture p0. First, p0 is deinterleaved into two fields p0Top and p0Bottom whose heights are equal to h0 >> 1. Then the chroma downsampling filtering process in the vertical direction is applied to p0Top and p0Bottom. Finally, the output fields of the filtering process p1Top and p1Bottom are interleaved to form the cropped output frame picture after vertical chroma downsampling.

* When chroma\_format\_idc is equal to 3 and target\_format\_idc is equal to either 1 or 2, the chroma downsampling filtering process in the horizontal direction is applied as follows:

divUh = 0  
for( j = 0; j < lenDh; j++ )  
 divDh += fDh[ 0 ][ j ]   
h0 = ( pic\_height\_in\_luma\_samples / SubHeightC ) – ( conf\_win\_top\_offset + conf\_win\_bottom\_offset )  
w0 = pic\_width\_in\_luma\_samples – ( conf\_win\_right\_offset + conf\_win\_left\_offset )  
for( v = 0; v < h0; v++ )  
 for( u = 0; u < ( w0 >> 1 ); u++ ) {  
 sum = 0  
 for( i = − ( lenDh − 1 ) / 2; i  <=  lenDh / 2; i++ )  
 sum += p0[ Clip3( 0, w0 − 1, ( u  <<  1 ) + i ) ][ v ] \* fDh[ 0 ][ i + ( lenDh − 1 ) / 2 ]  
 p1[ u ][ v ] = ( sum + ( divDh >> 1 ) ) / divDh  
 } (D‑6)

where p0[ ][ ] is the array of chroma samples in a cropped output picture before horizontal chroma downsampling, and p1[ ][ ] is the array of chroma samples in a cropped output picture after horizontal chroma downsampling.

Table D‑9 – Usage of chroma filter in the vertical direction

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| chromaSampleLocType | ver\_filtering\_field\_process\_flag | num\_vertical\_filters (when applicable) | upsamplingFlag | bottomFlag | Filter coefficients (fDv for downsamping or fUv for upsampling) | Range of j | Filter tap length (lenDv for downsampling or lenUv for upsampling) |
| 0, 1 | 0 | 2 | 0 | – | fDv[ 0 ][ j ] = verFilterCoeff[ 0 ][ j ] | 0..lenDv[ 0 ] - 1 | lenDv[ 0 ] =  verTapLength[ 0 ] |
| 1 | – | fUv[ 0 ][ j ] = verFilterCoeff[ 1 ][ j ] | 0..lenUv[ 0 ] - 1 | lenUv[ 0 ] =  verTapLength[ 1 ] |
| fUv[ 1 ][ j ] = verFilterCoeff[ 1 ][ verTapLength[ 1 ] +1 − j ] | 0..lenUv[ 1 ] - 1 | lenUv[ 1 ] =  verTapLength[ 1 ] |
| 1 | 3 | 0 | 0 | fDv[ 0 ][ j ] = verFilterCoeff[ 0 ][ j ] | 0..lenDv[ 0 ] - 1 | lenDv[ 0 ] = verTapLength[ 0 ] |
| 1 | fDv[ 1 ][ j ] = verFilterCoeff[ 0 ][ verTapLength[ 0 ] +1 − j ] | 0..lenDv[ 1 ] - 1 | lenDv[ 1 ] = verTapLength[ 0 ] |
| 1 | 0 | fUv[ 0 ][ j ] = verFilterCoeff[ 1 ][ j ] | 0..lenUv[ 0 ] - 1 | lenUv[ 0 ] = verTapLength[ 1 ] |
| fUv[ 1 ][ j ] = verFilterCoeff[ 2 ][ j ] | 0..lenUv[ 1 ] - 1 | lenUv[ 1 ] = verTapLength[ 2 ] |
| 1 | fUv[ 2 ][ j ] = verFilterCoeff[ 1 ][ verTapLength[ 1 ] +1 - j ] | 0..lenUv[ 2 ] - 1 | lenUv[ 2 ] =  verTapLength[ 1 ] |
| fUv[ 3 ][ j ] = verFilterCoeff[ 2 ][ verTapLength[ 2 ] +1 − j ] | 0..lenUv[ 3 ] - 1 | lenUv[ 3 ] = verTapLength[ 2 ] |
| 2, 3 | 0 | 3 | 0 | – | fDv[ 0 ][ j ] = verFilterCoeff[ 0 ][ j ] | 0..lenDv[ 0 ] - 1 | lenDv[ 0 ] = verTapLength[ 0 ] |
| 1 | – | fUv[ 0 ][ j ] = verFilterCoeff[ 1 ][ j ] | 0..lenUv[ 0 ] - 1 | lenUv[ 0 ] = verTapLength[ 1 ] |
| fUv[ 1 ][ j ] = verFilterCoeff[ 2 ][ j ] | 0..lenUv[ 1 ] - 1 | lenUv[ 1 ] = verTapLength[ 2 ] |
| 1 | 5 | 0 | 0 | fDv[ 0 ][ j ] = verFilterCoeff[ 0 ][ j ] | 0..lenDv[ 0 ] - 1 | lenDv[ 0 ] = verTapLength[ 0 ] |
| 1 | fDv[ 1 ][ j ] = verFilterCoeff[ 1 ][ j ]  with j = 0..verTapLength[ 1 ] +1 | 0..lenDv[ 1 ] - 1 | lenDv[ 1 ] = verTapLength[ 1 ] |
| 1 | 0 | fUv[ 0 ][ j ] = verFilterCoeff[ 2 ][ j ] | 0..lenUv[ 0 ] - 1 | lenUv[ 0 ] = verTapLength[ 2 ] |
| fUv[ 1 ][ j ] = verFilterCoeff[ 3 ][ j ] | 0..lenUv[ 1 ] - 1 | lenUv[ 1 ] = verTapLength[ 3 ] |
| 1 | fUv[ 2 ][ j ] = verFilterCoeff[ 4 ][ j ] | 0..lenUv[ 2 ] – 1 | lenUv[ 2 ] = verTapLength[ 4 ] |
| fUv[ 3 ][ j ] = verFilterCoeff[ 4 ][ verTapLength[ 4 ] +1 − i ] | 0..lenUv[ 3 ] – 1 | lenUv[ 3 ] = verTapLength[ 4 ] |
| 4, 5 | 0 | 3 | 0 | – | fDv[ 0 ][ j ] = verFilterCoeff[ 0 ][ j ] | 0..lenDv[ 0 ] – 1 | lenDv[ 0 ] = verTapLength[ 0 ] |
| 1 | – | fUv[ 0 ][ j ] = verFilterCoeff[ 1 ][ j ] | 0..lenUv[ 0 ] – 1 | lenUv[ 0 ] = verTapLength[ 1 ] |
| fUv[ 1 ][ j ] = verFilterCoeff[ 2 ][ j ] | 0..lenUv[ 1 ] – 1 | lenUv[ 1 ] = verTapLength[ 2 ] |
| 1 | 5 | 0 | 0 | fDv[ 0 ][ j ] = verFilterCoeff[ 0 ][ j ] | 0..lenDv[ 0 ] – 1 | lenDv[ 0 ] = verTapLength[ 0 ] |
| 1 | fDv[ 1 ][ j ] = verFilterCoeff[ 1 ][ j ] | 0..lenDv[ 1 ] – 1 | lenDv[ 1 ] = verTapLength[ 1 ] |
| 1 | 0 | fUv[ 0 ][ j ] = verFilterCoeff[ 2 ][ j ] | 0..lenUv[ 0 ] – 1 | lenUv[ 0 ] = verTapLength[ 2 ] |
| fUv[ 1 ][ j ] = verFilterCoeff[ 2 ][ verTapLength[ 2 ] + 1 − j ] | 0..lenUv[ 1 ] – 1 | lenUv[ 1 ] = verTapLength[ 2 ] |
| 1 | fUv[ 2 ][ j ] = verFilterCoeff[ 3 ][ j ] | 0..lenUv[ 2 ] – 1 | lenUv[ 2 ] = verTapLength[ 3 ] |
| fUv[ 3 ][ j ] = verFilterCoeff[ 4 ][ j ] | 0..lenUv[ 3 ] – 1 | lenUv[ 3 ] = verTapLength[ 4 ] |

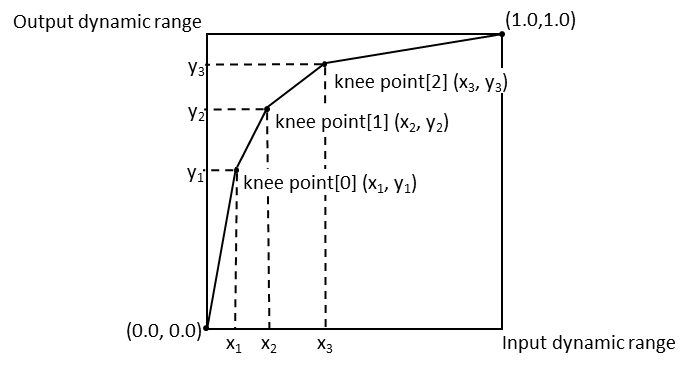
Table D‑10 – Usage of chroma filter in the horizontal direction

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| chromaSampleLocType | num\_horizontal\_filters (when applicable) | upsamplingFlag | Filter coefficients (fDh for downsamping or fUh for upsampling) | Range of j | Filter tap length (lenDh for downsampling or lenUh for upsampling) |
| 0,2,4 | 3 | 0 | fDh[ 0 ][ j ] = horFilterCoeff[ 0 ][ j ] | 0..lenDh[ 0 ] – 1 | lenDh = horTapLength[ 0 ] |
| 1 | fUh[ 0 ][ j ] = horFilterCoeff[ 1 ][ j ] | 0..lenUh[ 0 ] – 1 | lenUh[ 0 ] = horTapLength[ 1 ] |
| fUh[ 1 ][ j ] = horFilterCoeff[ 2 ][ j ] | 0..lenUh[ 1 ] – 1 | lenUh[ 1 ] = horTapLength[ 2 ] |
| 1,3,5 | 2 | 0 | fDh[ 0 ][ j ] = horFilterCoeff[ 0 ][ j ] | 0..lenDh[ 0 ] – 1 | lenDh = horTapLength[ 0 ] |
| 1 | fUh[ 0 ][ j ] = horFilterCoeff[ 1 ][ j ] | 0..lenUh[ 0 ] – 1 | lenUh[ 0 ] = horTapLength[ 1 ] |
| fUh[ 1 ][ j ] = horFilterCoeff[ 1 ][ horTapLength[ 1 ] − j ] | 0..lenUh[ 1 ] – 1 | lenUh[ 1 ] = horTapLength[ 1 ] |

D.3.27 Knee function information SEI message semantics

This SEI message provides information to enable decompression or compression of the colour samples of the output decoded pictures by using a knee function for customisation to particular display environments. The decompression or compression process maps the white level of sample values in the normalized linear RGB colour space. The compression or decompression should be applied to each RGB component produced by colour space conversion of the decoded image accordingly. [Ed. (JB): Consider defining what is meant here by “compression” or “decompression”]

A knee function is composed of several lines which starts with the point ( 0.0, 0.0 ), ends with the point ( 1.0, 1.0 ) and connects all knee points in ascending order of index i. The coordinate of the i-th knee point is specified by ( input\_knee\_point[ i ], output\_knee\_point[ i ] ). An example of knee function is shown in Figure D‑1.



**Figure D**‑**1** – **Knee Function**

**knee\_function\_id** contains an identifying number that may be used to identify the purpose of the knee functions. The value of knee\_function\_id shall be in the range of 0 to 232 − 2, inclusive.

Values of knee\_function\_id from 0 to 255 and from 512 to 231 − 1 may be used as determined by the application. Values of knee\_function\_id from 256 to 511, inclusive, and from 231 to 232 − 2, inclusive are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all knee function information SEI messages containing a value of knee\_function\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, and bitstreams shall not contain such values.

NOTE 1 – The knee\_function\_id can be used to support knee function process that are suitable for different display scenarios. For example, different values of knee\_function\_id may correspond to different display bit depths.

**knee\_function\_cancel\_flag** equal to 1 indicates that the knee function information SEI message cancels the persistence of any previous knee function information SEI message in output order. knee\_function\_cancel\_flag equal to 0 indicates that knee function information follows.

**knee\_function\_persistence\_flag** specifies the persistence of the knee function information SEI message.

knee\_function\_persistence\_flag equal to 0 specifies that the knee function information applies to the current decoded picture only.

knee\_function\_persistence\_flag equal to 1 specifies that the knee function information persists in output order until any of the following conditions are true:

– A new CVS begins.

– A picture in an access unit containing a knee function information SEI message with the same value of knee\_function\_id is output having PicOrderCntVal greater than PicOrderCnt( CurrPic ).

**mapping\_flag** equals to 0 indicates that the knee function is a decompression function. mapping\_flag equals to 1 indicates that the knee function is a compression function.

**input\_d\_range** specifies the peak luminance level for the input picture of the knee function process relative to the nominal luminance level in units of 0.1%. When the value of input\_d\_range is 0, the peak luminance level of the input picture is unspecified.

**input\_disp\_luminance** specifies the expected display brightness of peak luminance level for the input picture of the knee function process. The value of input\_disp\_luminance is in units of candela per square metre. When the value of input\_disp\_luminance is 0, the expected display brightness of peak luminance level for the input picture is unspecified.

**output\_d\_range** specifies the peak luminance level for the output picture of the knee function process relative to the nominal luminance level in units of 0.1%. When the value of output\_d\_range is 0, the peak luminance level of the output picture is unspecified.

**output\_disp\_luminance** specifies the expected display brightness of peak luminance level for the output picture of the knee function process. The value of output\_disp\_luminance is in units of candela per square metre. When the value of output\_disp\_luminance is 0, the expected display brightness of peak luminance level for the output picture is unspecified.

**num\_knee\_points\_minus1** plus 1 specifies the number of knee points. num\_knee\_points\_minus1 shall be in the range of 0 to 998, inclusive.

**input\_knee\_point**[ i ] specifies the luminance level of the i-th knee point of the input picture. The luminance level of the knee point of the input picture is normalized to the range of 0 to 1.0 in units of 0.1%. The value of input\_knee\_point shall be in the range of 1 to 999, inclusive. The value of the i-th input\_knee\_point shall be greater than the value of the (i−1)-th input\_knee\_point.

**output\_knee\_point**[ i ] specifies the luminance level of the i-th knee of the output picture. The luminance level of the knee point of the output picture is normalized to the range of 0 to 1.0 in units of 0.1%. The value of output\_knee\_point shall be in the range of 0 to 1000, inclusive.

NOTE 2 – The input dynamic range (in percent) may be derived using input\_d\_range \* input\_knee\_point[ i ] ÷ 10000. The output dynamic range (in percent) may be derived using output\_d\_range \* output\_knee\_point[ i ] ÷ 10000.

# References

[1] JCTVC-Q1008 “Scalable extensions” , June 2014

[2] Society of Motion Pictures and Television Engineers, “4:2:2 / 4:2:0 Format Conversion Minimizing Color Difference Signal Degradation in Concatenated Operations — Filtering” [RP 2050-1:2012](http://standards.smpte.org/content/978-1-61482-653-8/rp-2050-1-2012/SEC1)

# Patent rights declaration(s)

**MovieLabs does not have any current or pending patent rights relating to the technology described in this contribution.**