|  |
| --- |
| **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ORGANISATION INTERNATIONALE DE NORMALISATION ISO/IEC JTC 1/SC 29/WG 5 MPEG JOINT VIDEO CODING TEAM WITH ITU-T SG 16** |
| **ISO/IEC JTC 1 / SC 29 / WG 5 N 126** |
| **Online, 20–29 April 2022** |
| |  |  | | --- | --- | | **Title:** | **Working draft of ISO/IEC 23002-7:202x (2nd Ed.) Amd.1 Additional SEI messages** | | **Source:** | **Convenor (Jens-Rainer Ohm)** | | **Type:** | **Project** | | **Subtype:** | **Draft** | | **Status:** | **Approved** | | **Date:** | **2022-06-21** | | **Expected Action:** | **Info** | | **Action due date:** | **N/A** | | **No. of pages** | **16** (without this cover page) | | **Email of convenor:** | **ohm @ ient . rwth-aachen . de** | | **Committee URL:** | **https://sd.iso.org/documents/ui/#!/browse/iso/iso-iec-jtc-1/iso-iec-jtc-1-sc-29/iso-iec-jtc-1-sc-29-wg-5** | |

|  |  |
| --- | --- |
| **Joint Video Experts Team (JVET)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29**  26th Meeting, by teleconference, 20–29 April 2022 | Document: JVET-Z2006-v1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Additional SEI messages for VSEI (Draft 1)** | | |
| *Status:* | Output document approved by JVET | | |
| *Purpose:* | Draft text | | |
| *Author(s) or Contact(s):* | Sean McCarthy Takeshi Chujoh Miska Hannuksela Gary Sullivan Ye-Kui Wang | Email: | sean.mccarthy@dolby chujoh.takeshi@sharp.co.jp miska.hannuksela@nokia.com garysull@microsoft.com yekui.wang@bytedance.com |
| *Source:* | Editors | | |

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

# Abstract

This document contains the draft text for changes to the versatile supplemental enhancement information messages for coded video bitstreams (VSEI) standard (Rec. ITU-T H.274 | ISO/IEC 23002-7), to specify additional SEI messages, including the shutter interval information SEI message and SEI messages to facilitate neural-network-based post-processing filtering.

**Changes to the specification text:**

*In clause 2.3, add the following references:*

– IEEE 754-2019, *IEEE Standard for Floating-Point Arithmetic*.

– ISO/IEC 15938-17, *Information technology – Multimedia content description interface – Part 17: Compression of neural networks for multimedia content description and analysis.*

*In clause 5.8, add the following function definition immediately before the definition of the function Round( x ) and adjust equation indices accordingly:*

Reflect( y, z ) = (12)

*In clauses 8.1, replace Table 4 with the following:*

|  |  |
| --- | --- |
| SEI message | Persistence scope |
| Filler payload | The PU containing the SEI message |
| User data registered by Rec. ITU-T T.35 | Unspecified |
| User data unregistered | Unspecified |
| Film grain characteristics | Specified by the syntax of the SEI message |
| Frame packing arrangement | Specified by the syntax of the SEI message |
| Display orientation | Specified by the syntax of the SEI message |
| Referenced parameter sets | The CLVS containing the SEI message |
| Decoded picture hash | The PU containing the SEI message |
| Mastering display colour volume | The CLVS containing the SEI message |
| Colour transform information | Specified by the syntax of the SEI message |
| Content light level information | The CLVS containing the SEI message |
| DRAP indication | The picture associated with the SEI message |
| Alternative transfer characteristics | The CLVS containing the SEI message |
| Ambient viewing environment | The CLVS containing the SEI message |
| Content colour volume | Specified by the syntax of the SEI message |
| Equirectangular projection | Specified by the syntax of the SEI message |
| Generalized cubemap projection | Specified by the syntax of the SEI message |
| Sphere rotation | Specified by the syntax of the SEI message |
| Region-wise packing | Specified by the syntax of the SEI message |
| Omnidirectional viewport | Specified by the syntax of the SEI message |
| Alpha channel information | Specified by the syntax of the SEI message |
| Frame-field information | The PU containing the SEI message |
| Depth representation information | Specified by the semantics of the SEI message |
| Multiview acquisition information | The CVS containing the SEI message |
| Multiview view position | The CVS containing the SEI message |
| Annotated regions | Specified by the syntax of the SEI message |
| Sample aspect ratio information | Specified by the syntax of the SEI message |
| Scalability dimension information | The CVS containing the SEI message |
| Extended DRAP indication | The picture associated with the SEI message |
| Shutter interval information | The CVS containing the SEI message |
| Neural-network post-filter characteristics | The CVS containing the SEI message |
| Neural-network post-filter activation | The PU containing the SEI message |

*Add clauses 8.27 to 8.29 as follows:*

* 1. **Shutter interval information SEI message**

8.27.1 Shutter interval information SEI message syntax

|  |  |
| --- | --- |
| shutter\_interval\_info( payloadSize ) { | **Descriptor** |
| **sii\_time\_scale** | u(32) |
| **fixed\_shutter\_interval\_within\_clvs\_flag** | u(1) |
| if( fixed\_shutter\_interval\_within\_clvs\_flag ) |  |
| **sii\_num\_units\_in\_shutter\_interval** | u(32) |
| else { |  |
| **sii\_max\_sub\_layers\_minus1** | u(3) |
| for( i = 0; i <= sii\_max\_sub\_layers\_minus1; i++ ) |  |
| **sub\_layer\_num\_units\_in\_shutter\_interval**[ i ] | u(32) |
| } |  |
| } |  |

8.27.2 Shutter interval information SEI message syntax

The shutter interval information SEI message indicates the shutter interval for the associated video source pictures prior to encoding, e.g., for camera-captured content, the shutter interval is amount of time that an image sensor is exposed to produce each source picture.

When a shutter interval information SEI message is present for any picture of a CLVS of a particular layer, a shutter interval information SEI message shall be present for the first picture of the CLVS. The shutter interval information SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All shutter interval information SEI messages that apply to the same CLVS shall have the same content.

**sii\_time\_scale** specifies the number of time units that pass in one second. The value of sii\_time\_scale shall be greater than 0. For example, a time coordinate system that measures time using a 27 MHz clock has an sii\_time\_scale of 27 000 000.

**fixed\_shutter\_interval\_within\_clvs\_flag** equal to 1 specifies that the indicated shutter interval is the same for all temporal sublayers in the CLVS. fixed\_shutter\_interval\_within\_clvs\_flagequal to 0 specifies that the indicated shutter interval may not be the same for all temporal sublayers in the CLVS.

**sii\_num\_units\_in\_shutter\_interval**, when fixed\_shutter\_interval\_within\_clvs\_flag is equal to 1, specifies the number of time units of a clock operating at the frequency sii\_time\_scale Hz that corresponds to the indicated shutter interval of each picture in the CLVS. The value 0 may be used to indicate that the associated video content contains screen capture content, computer generated content, or other non-camera-captured content.

The indicated shutter interval, denoted by the variable shutterInterval, in units of seconds, is equal to the quotient of sii\_num\_units\_in\_shutter\_interval divided by sii\_time\_scale. For example, to represent a shutter interval equal to 0.04 seconds, sii\_time\_scale may be equal to 27 000 000 and sii\_num\_units\_in\_shutter\_interval may be equal to 1 080 000.

**sii\_max\_sub\_layers\_minus1** plus 1 specifies the maximum number of temporal sublayers that may be present in each CLVS referring to the SPS.

NOTE – For example, the information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of ATSC A/341:2022-03 Annex D when sii\_max\_sub\_layers\_minus1 is equal to 1 and fixed\_shutter\_interval\_within\_clvs\_flag is equal to 0.

**sub\_layer\_num\_units\_in\_shutter\_interval**[ i ], when present, specifies the number of time units of a clock operating at the frequency sii\_time\_scale Hz that corresponds to the shutter interval of each picture with temporal sublayer identifier equal to i in the CLVS. The shutter interval for each picture with temporal sublayer identifier equal to i in the CLVS, denoted by the variable subLayerShutterInterval[ i ], in units of seconds, is equal to the quotient of sub\_layer\_num\_units\_in\_shutter\_interval[ i ] divided by sii\_time\_scale.

The variable subLayerShutterInterval[ i ], corresponding to the indicated shutter interval of each picture with temporal sublayer identifier equal to i in the CLVS, is thus derived as follows:

if( fixed\_shutter\_interval\_within\_clvs\_flag )  
 subLayerShutterInterval[ i ] = sii\_num\_units\_in\_shutter\_interval ÷ sii\_time\_scale (72)  
else  
 subLayerShutterInterval[ i ] = sub\_layer\_num\_units\_in\_shutter\_interval[ i ] ÷ sii\_time\_scale

* 1. **Neural-network post-filter characteristics SEI message**

8.28.1 Neural-network post-filter characteristics SEI message syntax

|  |  |
| --- | --- |
| nn\_post\_filter\_characteristics( payloadSize ) { | **Descriptor** |
| **nnpfc\_id** | ue(v) |
| **nnpfc\_mode\_idc** | ue(v) |
| if( nnpfc\_mode\_idc = = 1 ) { |  |
| **nnpfc\_purpose** | ue(v) |
| if( nnpfc\_purpose = = 2 | | nnpfc\_purpose = = 4 ) { |  |
| **nnpfc\_out\_sub\_width\_c\_flag** | u(1) |
| **nnpfc\_out\_sub\_height\_c\_flag** | u(1) |
| } |  |
| if( nnpfc\_purpose = = 3 | | nnpfc\_purpose = = 4 ) { |  |
| **nnpfc\_pic\_width\_in\_luma\_samples** | ue(v) |
| **nnpfc\_pic\_height\_in\_luma\_samples** | ue(v) |
| } |  |
| /\* input and output formatting \*/ |  |
| **nnpfc\_component\_last\_flag** | u(1) |
| **nnpfc\_inp\_sample\_idc** | ue(v) |
| if( nnpfc\_inp\_sample\_idc = = 4 ) |  |
| **nnpfc\_inp\_tensor\_bitdepth\_minus8** | ue(v) |
| **nnpfc\_inp\_order\_idc** | ue(v) |
| **nnpfc\_out\_sample\_idc** | ue(v) |
| if( nnpfc\_out\_sample\_idc = = 4 ) |  |
| **nnpfc\_out\_tensor\_bitdepth\_minus8** | ue(v) |
| **nnpfc\_out\_order\_idc** | ue(v) |
| **nnpfc\_constant\_patch\_size\_flag** | u(1) |
| **nnpfc\_patch\_width\_minus1** | ue(v) |
| **nnpfc\_patch\_height\_minus1** | ue(v) |
| **nnpfc\_overlap** | ue(v) |
| **nnpfc\_padding\_type** | ue(v) |
| **nnpfc\_complexity\_idc** | ue(v) |
| if( nnpfc\_complexity\_idc > 0 ) |  |
| nnpfc\_complexity\_element( nnpfc\_complexity\_idc ) |  |
| } |  |
| /\* filter specified or updated by ISO/IEC 15938-17 bitstream \*/ |  |
| if( nnpfc\_mode\_idc = = 1 ) { |  |
| while( !byte\_aligned( ) ) |  |
| **nnpfc\_reserved\_zero\_bit** | u(1) |
| for( i = 0; more\_data\_in\_payload( ); i++ ) |  |
| **nnpfc\_payload\_byte**[ i ] | b(8) |
| } |  |
| } |  |

|  |  |
| --- | --- |
| nnpfc\_complexity\_element( nnpfc\_complexity\_idc ) { | **Descriptor** |
| if( nnpfc\_complexity\_idc = = 1 ) { |  |
| **nnpfc\_parameter\_type\_flag** | u(1) |
| **nnpfc\_log2\_parameter\_bit\_length\_minus3** | u(2) |
| **nnpfc\_num\_parameters\_idc** | u(8) |
| **nnpfc\_num\_kmac\_operations\_idc** | ue(v) |
| } |  |
| } |  |

* + 1. Neural-network post-filter characteristics SEI message semantics

This SEI message specifies a neural network that may be used as a post-processing filter. The use of specified post-processing filters for specific pictures is indicated with neural-network post-filter activation SEI messages.

Use of this SEI message requires the definition of the following variables:

– Cropped decoded output picture width and height in units of luma samples, denoted herein by InpPicWidthInLumaSamples and InpPicHeightInLumaSamples, respectively.

– Luma sample array CroppedYPic[ y ][ x ] and chroma sample arrays CroppedCbPic[ y ][ x ] and CroppedCrPic[ y ][ x ], when present, of the cropped decoded output picture for vertical coordinates y and horizontal coordinates x, where the top-left corner of the sample array has coordinates y equal to 0 and x equal to 0.

– Bit depth BitDepthY for the luma sample array of the cropped decoded output picture.

– Bit depth BitDepthC for the chroma sample arrays, if any, of the cropped decoded output picture.

– Chroma subsampling ratio relative to luma denoted as InpSubWidthC and InpSubHeightC.

– When nnpfc\_inp\_order\_idc is equal to 3, the initial luma quantization parameter value SliceQPY.

When this SEI message specifies a neural network that may be used as a post-processing filter, the semantics specify the derivation of the luma sample array FilteredYPic[ y ][ x ] and chroma sample arrays FilteredCbPic[ y ][ x ] and FilteredCrPic[ y ][ x ], as indicated by the value of nnpfc\_out\_order\_idc, that contain the output of the post-processing filter.

**nnpfc\_id** contains an identifying number that may be used to identify a post-processing filter. The value of nnpfc\_id shall be in the range of 0 to 232 − 2, inclusive.

Values of nnpfc\_id from 256 to 511, inclusive, and from 231 to 232 − 2, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of nnpfc\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, shall ignore it.

**nnpfc\_mode\_idc** equal to 0 specifies that the post-processing filter associated with the nnpfc\_id value is determined by external means not specified in this Specification.

nnpfc\_mode\_idc equal to 1 specifies that the post-processing filter associated with the nnpfc\_id value is a neural network represented by the ISO/IEC 15938-17 bitstream contained in this SEI message.

The value of nnpfc\_mode\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_mode\_idc greater than 1 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore SEI messages that contain reserved values of nnpfc\_mode\_idc.

When the current CLVS contains a preceding neural-network post-filter characteristics SEI message, in decoding order, that has the same value of nnpfc\_id equal to the value of nnpfc\_id in this SEI message, at least one of the following conditions shall apply:

– This SEI message has nnpfc\_mode\_idc equal to 1 and the same content as the preceding neural-network post-filter characteristics SEI message except that the values of nnpfc\_mode\_idc and nnpfc\_payload\_byte[ i ] may differ in order to provide a neural network update.

– This SEI message has the same content as the preceding neural-network post-filter characteristics SEI message.

When this SEI message is the first neural-network post-filter characteristics SEI message, in decoding order, that has a particular nnpfc\_id value within the current CLVS, it specifies a base post-processing filter that pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS. When this SEI message is not the first neural-network post-filter characteristics SEI message, in decoding order, that has a a particular nnpfc\_id value within the current CLVS, this SEI message pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS or the next neural-network post-filter characteristics SEI message having that particular nnpfc\_id value, in output order, within the current CLVS.

**nnpfc\_purpose** indicates the purpose of post-processing filter as specified in Table 20. The value of nnpfc\_purpose shall be in the range of 0 to 232 − 2, inclusive. Values of nnpfc\_purpose that do not appear in Table 20 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore SEI messages that contain reserved values of nnpfc\_purpose.

**Table 20 – Definition of nnpfc\_purpose**

|  |  |
| --- | --- |
| **Value** | **Interpretation** |
| 0 | Unknown or unspecified |
| 1 | Visual quality improvement |
| 2 | Chroma upsampling from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format |
| 3 | Increasing the width or height of the cropped decoded output picture without changing the chroma format |
| 4 | Increasing the width or height of the cropped decoded output picture and upsampling the chroma format |

NOTE 1 – When a reserved value of nnpfc\_purpose is taken into use in the future by ITU-T | ISO/IEC, the syntax of this SEI message could be extended with syntax elements whose presence is conditioned by nnpfc\_purpose being equal to that value.

**nnpfc\_out\_sub\_width\_c\_flag** and **nnpfc\_out\_sub\_height\_c\_flag** are used to derive the variables outSubWidthC and outSubHeightC, respectively, which specify the chroma subsampling ratio relative to luma in the picture resulting from the post-processing filtering. When not present, nnpfc\_out\_sub\_width\_c\_flag and nnpfc\_out\_sub\_height\_c\_flag are both inferred to be equal to 0. When nnpfc\_out\_sub\_width\_c\_flag and nnpfc\_out\_sub\_height\_c\_flag are present, the sum of nnpfc\_out\_sub\_width\_c\_flag and nnpfc\_out\_sub\_height\_c\_flag shall be greater than 0.

outSubWidthC = InpSubWidthC − nnpfc\_out\_sub\_width\_c\_flag(73)

outSubHeightC = InpSubHeightC − nnpfc\_out\_sub\_height\_c\_flag(74)

It is a requirement of bitstream conformance that outSubWidthC and outSubHeightC are both greater than 0.

**nnpfc\_pic\_width\_in\_luma\_samples** and **nnpfc\_pic\_height\_in\_luma\_samples** specify the width and height, respectively, of the luma sample array of the picture resulting by applying the post-processing filter identified by nnpfc\_id to a cropped decoded output picture. When nnpfc\_pic\_width\_in\_luma\_samples and nnpfc\_pic\_height\_in\_luma\_samples are not present, they are inferred to be equal to InpPicWidthInLumaSamples and InpPicHeightInLumaSamples, respectively.

**nnpfc\_component\_last\_flag** equal to 0 specifies that the second dimension in the input tensor inputTensor to the post-processing filter and the output tensor outputTensor resulting from the post-processing filter is used for the channel. nnpfc\_component\_last\_flag equal to 1 specifies that the last dimension in the input tensor inputTensor to the post-processing filter and the output tensor outputTensor resulting from the post-processing filter is used for the channel.

NOTE 2 – The specified values of nnpfc\_inp\_sample\_idc and nnpfc\_out\_sample\_idc specify that the first dimension in the input tensor and in the output tensor, respectively, is used for the batch index, which is a practice in some neural network frameworks. While the semantics of this SEI message use batch size equal to 1, it is up to the post-processing implementation to determine the batch size used as input to the neural network inference.

NOTE 3 – A colour component is an example of a channel.

**nnpfc\_inp\_sample\_idc** indicates the method of converting a sample value of the cropped decoded output picture to an input value to the post-processing filter. When nnpfc\_inp\_sample\_idc is equal to 0, 1, 2, or 3, the input values to the post-processing filter are binary16, binary32, binary64, or binary128 floating point values, respectively, as specified in IEEE 754-2019, and the functions InpY and InpC are specified as follows:

InpY( x ) = x ÷ ( ( 1  <<  BitDepthY ) − 1 )(75)

InpC( x )= x ÷ ( ( 1  <<  BitDepthC ) − 1 )(76)

When nnpfc\_inp\_sample\_idc is equal to 4, the input values to the post-processing filter are unsigned integer and the functions InpY and InpC are specified as follows:

shift = BitDepthY − inpTensorBitDepth  
if( inpTensorBitDepth >= BitDepthY)  
 InpY( x ) = x  <<  ( inpTensorBitDepth − BitDepthY )  
else  
 InpY( x ) = Clip3(0, ( 1  <<  inpTensorBitDepth ) − 1, ( x + ( 1  <<  ( shift − 1 ) ) )  >>  shift ) (77)  
shift = BitDepthC − inpTensorBitDepth  
if( inpTensorBitDepth >= BitDepthC)  
 InpC( x ) = x  <<  ( inpTensorBitDepth − BitDepthC )  
else  
 InpC( x ) = Clip3(0, ( 1  <<  inpTensorBitDepth ) − 1, ( x + ( 1  <<  ( shift − 1 ) ) )  >>  shift )

The variable inpTensorBitDepth is derived from the syntax element nnpfc\_inp\_tensor\_bitdepth\_minus8 as specified below.

The value of nnpfc\_inp\_sample\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_inp\_sample\_idc greater than 4 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore SEI messages that contain reserved values of nnpfc\_inp\_sample\_idc.

**nnpfc\_inp\_tensor\_bitdepth\_minus8** plus 8 specifies the bit depth of luma sample values in the input integer tensor. The value of inpTensorBitDepth is derived as follows:

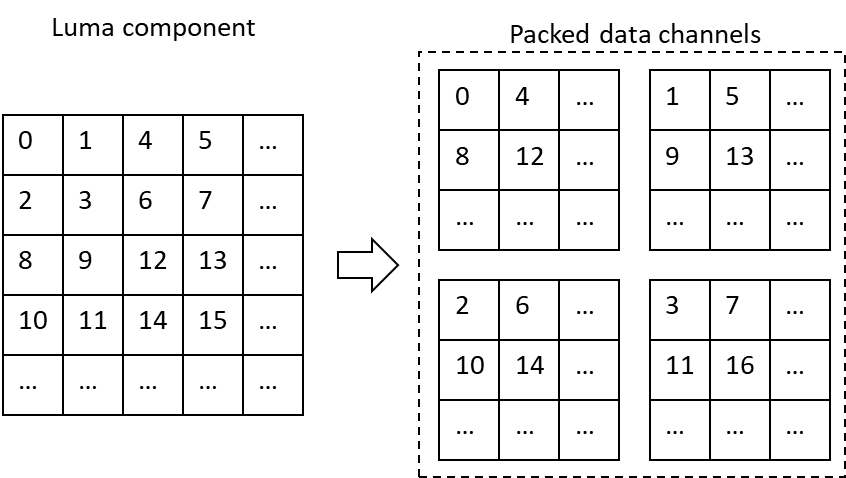
inpTensorBitDepth = nnpfc\_inp\_tensor\_bitdepth\_minus8 + 8(78)

It is a requirement of bitstream conformance that the value of nnpfc\_inp\_tensor\_bitdepth\_minus8 shall be in the range of 0 to 24, inclusive.

**nnpfc\_inp\_order\_idc** indicates the method of ordering the sample arrays of a cropped decoded output picture as the input to the post-processing filter. Table 21 contains an informative description of nnpfc\_inp\_order\_idc values. The semantics of nnpfc\_inp\_order\_idc in the range of 0 to 3, inclusive, are specified in Table 23, which specifies a process for deriving the input tensors inputTensor for different values of nnpfc\_inp\_order\_idc and a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensors. When the chroma format of the cropped decoded output picture is not 4:2:0, nnpfc\_inp\_order\_idc shall not be equal to 3. The value of nnpfc\_inp\_order\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_inp\_order\_idc greater than 3 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore SEI messages that contain reserved values of nnpfc\_inp\_order\_idc.

**Table 21 – Informative description of nnpfc\_inp\_order\_idc values**

|  |  |
| --- | --- |
| **nnpfc\_inp\_ order\_idc** | **Description** |
| 0 | Only the luma matrix is present in the input tensor, thus the number of channels is 1. |
| 1 | Only the chroma matrices are present in the input tensor, thus the number of channels is 2. |
| 2 | The luma and chroma matrices are present in the input tensor, thus the number of channels is 3. |
| 3 | Four luma matrices, two chroma matrices, and a quantization parameter matrix are present in the input tensor, thus the number of channels is 7. The luma channels are derived in an interleaved manner as illustrated in Figure 12. This nnpfc\_inp\_order\_idc can only be used when the chroma format is 4:2:0. |
| 4..255 | reserved |



**Figure 12 – Illustration of luma data channels of nnpfc\_inp\_order\_idc equal to 3 (informative)**

A patch is a rectangular array of samples from a component (e.g., a luma or chroma component) of a picture.

**nnpfc\_constant\_patch\_size\_flag** equal to 0 specifies that the post-processing filter accepts any patch size that is a positive integer multiple of the patch size indicated by nnpfc\_patch\_width\_minus1 and nnpfc\_patch\_height\_minus1 as input. nnpfc\_constant\_patch\_size\_flag equal to 1 specifies that the post-processing filter accepts exactly the patch size indicated by nnpfc\_patch\_width\_minus1 and nnpfc\_patch\_height\_minus1 as input.

**nnpfc\_patch\_width\_minus1** + 1, when nnpfc\_constant\_patch\_size\_flag equal to 1, specifies the horizontal sample counts of the patch size required for the input to the post-processing filter. When nnpfc\_constant\_patch\_size\_flag is equal to 0, any positive integer multiple of ( nnpfc\_patch\_width\_minus1 + 1 ) may be used as the horizontal sample counts of the patch size used for the input to the post-processing filter. The value of nnpfc\_patch\_width\_minus1 shall be in the range of 0 to 32766, inclusive.

**nnpfc\_patch\_height\_minus1** + 1, when nnpfc\_constant\_patch\_size\_flag equal to 1, specifies the vertical sample counts of the patch size required for the input to the post-processing filter. When nnpfc\_constant\_patch\_size\_flag is equal to 0, any positive integer multiple of ( nnpfc\_patch\_height\_minus1 + 1 ) may be used as the vertical sample counts of the patch size used for the input to the post-processing filter. The value of nnpfc\_patch\_height\_minus1 shall be in the range of 0 to 32766, inclusive.

**nnpfc\_overlap** specifies the overlapping horizontal and vertical sample counts of adjacent input tensors of the post-processing filter. The value of nnpfc\_overlap shall be in the range of 0 to 16383, inclusive.

The variables inpPatchWidth, inpPatchHeight, outPatchWidth, outPatchHeight, horCScaling, verCScaling, outPatchCWidth, outPatchCHeight, and overlapSize are derived as follows:

inpPatchWidth = nnpfc\_patch\_width\_minus1 + 1  
inpPatchHeight = nnpfc\_patch\_height\_minus1 + 1  
outPatchWidth = ( nnpfc\_pic\_width\_in\_luma\_samples \* inpPatchWidth ) / InpPicWidthInLumaSamples  
outPatchHeight = ( nnpfc\_pic\_height\_in\_luma\_samples \* inpPatchHeight ) / InpPicHeightInLumaSamples  
horCScaling = InpSubWidthC / outSubWidthC  
verCScaling = InpSubHeightC / outSubHeightC (79)  
outPatchCWidth = outPatchWidth \* horCScaling  
outPatchCHeight = outPatchHeight \* verCScaling  
overlapSize = nnpfc\_overlap

It is a requirement of bitstream conformance that outPatchWidth \* InpPicWidthInLumaSamples shall be equal to nnpfc\_pic\_width\_in\_luma\_samples \* inpPatchWidth and outPatchHeight \* InpPicHeightInLumaSamples shall be equal to nnpfc\_pic\_height\_in\_luma\_samples \* inpPatchHeight.

**nnpfc\_padding\_type** specifies the process of padding when referencing sample locations outside the boundaries of the cropped decoded output picture as described in Table 22. The value of nnpfc\_padding\_type shall be in the range of 0 to 15, inclusive.

**Table 22 – Informative description of nnpfc\_padding\_type values**

|  |  |
| --- | --- |
| **nnpfc\_padding\_type** | **Description** |
| 0 | zero padding |
| 1 | replication padding |
| 2 | reflection padding |
| 3..15 | reserved |

The function InpSampleVal( y, x, picHeight, picWidth, croppedPic ) with inputs being a vertical sample location y, a horizontal sample location x, a picture height picHeight, a picture width picWidth, and sample array croppedPic returns the value of sampleVal derived as follows:

if( nnpfc\_padding\_type = = 0 )  
 if( y < 0 | | x < 0 | | y >= picHeight | | x >= picWidth )  
 sampleVal = 0  
 else  
 sampleVal = croppedPic[ y ][ x ] (80)  
else if( nnpfc\_padding\_type = = 1 )  
 sampleVal = croppedPic[ Clip3( 0, picHeight − 1, y ) ][ Clip3( 0, picWidth − 1, x ) ]  
else /\* nnpfc\_padding\_type = = 2 \*/  
 sampleVal = croppedPic[ Reflect( picHeight − 1, y ) ][ Reflect( picWidth − 1, x ) ]

**Table 23 – Process for deriving the input tensors inputTensor for a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensors**

|  |  |
| --- | --- |
| **nnpfc\_inp\_ order\_idc** | **Process DeriveInputTensors( ) for deriving input tensors** |
| 0 | for( yP = −overlapSize; yP < inpPatchHeight + overlapSize; yP++)  for( xP = −overlapSize; xP < inpPatchWidth + overlapSize; xP++ ) {  inpVal = InpY( InpSampleVal( cTop + yP, cLeft + xP, InpPicHeightInLumaSamples,  InpPicWidthtInLumaSamples, CroppedYPic ) )  if( nnpfc\_component\_last\_flag = = 0 )  inputTensor[ 0 ][ 0 ][ yP + overlapSize ][ xP + overlapSize ] = inpVal  else  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 0 ] = inpVal  } |
| 1 | for( yP = −overlapSize; yP < inpPatchHeight + overlapSize; yP++)  for( xP = −overlapSize; xP < inpPatchWidth + overlapSize; xP++ ) {  inpCbVal = InpC( InpSampleVal( cTop + yP, cLeft + xP, InpPicHeightInLumaSamples / InpSubHeightC,  InpPicWidthtInLumaSamples / InpSubWidthC, CroppedCbPic ) )  inpCrVal = InpC( InpSampleVal( cTop + yP, cLeft + xP, InpPicHeightInLumaSamples / InpSubHeightC,  InpPicWidthtInLumaSamples / InpSubWidthC, CroppedCrPic ) )  if( nnpfc\_component\_last\_flag = = 0 ) {  inputTensor[ 0 ][ 0 ][ yP + overlapSize ][ xP + overlapSize ] = inpCbVal  inputTensor[ 0 ][ 1 ][ yP + overlapSize ][ xP + overlapSize ] = inpCrVal  } else {  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 0 ] = inpCbVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 1 ] = inpCrVal  }  } |
| 2 | for( yP = −overlapSize; yP < inpPatchHeight + overlapSize; yP++)  for( xP = −overlapSize; xP < inpPatchWidth + overlapSize; xP++ ) {  yY = cTop + yP  xY = cLeft + xP  yC = yY / InpSubHeightC  xC = xY / InpSubWidthC   inpYVal = InpY( InpSampleVal( yY, xY, InpPicHeightInLumaSamples,  InpPicWidthtInLumaSamples, CroppedYPic ) )  inpCbVal = InpC( InpSampleVal( yC, xC, InpPicHeightInLumaSamples / InpSubHeightC,  InpPicWidthtInLumaSamples / InpSubWidthC, CroppedCbPic ) )  inpCrVal = InpC( InpSampleVal( yC, xC, InpPicHeightInLumaSamples / InpSubHeightC,  InpPicWidthtInLumaSamples / InpSubWidthC, CroppedCrPic ) )  if( nnpfc\_component\_last\_flag = = 0 ) {  inputTensor[ 0 ][ 0 ][ yP + overlapSize ][ xP + overlapSize ] = inpYVal  inputTensor[ 0 ][ 1 ][ yP + overlapSize ][ xP + overlapSize ] = inpCbVal  inputTensor[ 0 ][ 2 ][ yP + overlapSize ][ xP + overlapSize ] = inpCrVal  } else {  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 0 ] = inpYVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 1 ] = inpCbVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 2 ] = inpCrVal  }  } |
| 3 | for( yP = −overlapSize; yP < inpPatchHeight + overlapSize; yP++)  for( xP = −overlapSize; xP < inpPatchWidth + overlapSize; xP++ ) {  yTL = cTop + yP \* 2  xTL = cLeft + xP \* 2  yBR = yTL + 1  xBR = xTL + 1  yC = cTop / 2 + yP  xC = cLeft / 2 + xP  inpTLVal = InpY( InpSampleVal( yTL, xTL, InpPicHeightInLumaSamples,  InpPicWidthtInLumaSamples, CroppedYPic ) )  inpTRVal = InpY( InpSampleVal( yTL, xBR, InpPicHeightInLumaSamples,  InpPicWidthtInLumaSamples, CroppedYPic ) )  inpBLVal = InpY( InpSampleVal( yBR, xTL, InpPicHeightInLumaSamples,  InpPicWidthtInLumaSamples, CroppedYPic ) )  inpBRVal = InpY( InpSampleVal( yBR, xBR, InpPicHeightInLumaSamples,  InpPicWidthtInLumaSamples, CroppedYPic ) )  inpCbVal = InpC( InpSampleVal( yC, xC, InpPicHeightInLumaSamples / 2,  InpPicWidthtInLumaSamples / 2, CroppedCbPic ) )  inpCrVal = InpC( InpSampleVal( yC, xC, InpPicHeightInLumaSamples / 2,  InpPicWidthtInLumaSamples / 2, CroppedCrPic ) )  if( nnpfc\_component\_last\_flag = = 0 ) {  inputTensor[ 0 ][ 0 ][ yP + overlapSize ][ xP + overlapSize ] = inpTLVal  inputTensor[ 0 ][ 1 ][ yP + overlapSize ][ xP + overlapSize ] = inpTRVal  inputTensor[ 0 ][ 2 ][ yP + overlapSize ][ xP + overlapSize ] = inpBLVal  inputTensor[ 0 ][ 3 ][ yP + overlapSize ][ xP + overlapSize ] = inpBRVal  inputTensor[ 0 ][ 4 ][ yP + overlapSize ][ xP + overlapSize ] = inpCbVal  inputTensor[ 0 ][ 5 ][ yP + overlapSize ][ xP + overlapSize ] = inpCrVal  inputTensor[ 0 ][ 6 ][ yP + overlapSize ][ xP + overlapSize ] = 2(SliceQPY – 42)/6  } else {  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 0 ] = inpTLVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 1 ] = inpTRVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 2 ] = inpBLVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 3 ] = inpBRVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 4 ] = inpCbVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 5 ] = inpCrVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 6 ] = 2(SliceQPY – 42)/6  }  } |
| 4..255 | reserved |

**nnpfc\_complexity\_idc** greater than 0 specifies that one or more syntax elements that indicate the complexity of the post-processing filter associated with the nnpfc\_id may be present. nnpfc\_complexity\_idc equal to 0 specifies that no syntax element that indicates the complexity of the post-processing filter associated with the nnpfc\_id is present. The value nnpfc\_complexity\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_complexity\_idc greater than 1 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore SEI messages that contain reserved values of nnpfc\_complexity\_idc.

**nnpfc\_out\_sample\_idc** equal to 0, 1, 2, or 3 indicates that the sample values output by the post-processing filter are binary16, binary32, binary64, or binary128 floating point values, respectively, as specified in IEEE 754-2019. Functions OutY and OutC for converting luma sample values and chroma sample values, respectively, output by the post-processing, to integer values at bit depths BitDepthY and BitDepthC, respectively, are specified as follows:

OutY( x ) = Clip3( 0, ( 1  <<  BitDepthY ) − 1, Round( x \* ( ( 1  <<  BitDepthY ) − 1 ) ) )(81)

OutC( x )= Clip3( 0, ( 1  <<  BitDepthC ) − 1, Round( x \* ( ( 1  <<  BitDepthC ) − 1 ) ) )(82)

nnpfc\_out\_sample\_idc equal to 4 indicates that the sample values output by the post-processing filter are unsigned integer. Functions OutY and OutC are specified as follows:

shift = outTensorBitDepth − BitDepthY  
if( outTensorBitDepth >= BitDepthY )  
 OutY( x ) = Clip3( 0, ( 1  <<  BitDepthY ) − 1, ( x + ( 1  <<  ( shift − 1 ) ) )  >>  shift )  
else  
 OutY( x ) = x  <<  ( BitDepthY − outTensorBitDepth ) (83)  
shift = outTensorBitDepth − BitDepthC  
if( outTensorBitDepth >= BitDepthC)  
 OutC( x )= Clip3( 0, ( 1  <<  BitDepthC ) − 1, ( x + ( 1  <<  ( shift − 1 ) ) )  >>  shift )  
else  
 OutC( x ) = x  <<  ( BitDepthC − outTensorBitDepth )

The variable outTensorBitDepth is derived from the syntax element nnpfc\_out\_tensor\_bitdepth\_minus8 as described below.

The value of nnpfc\_out\_sample\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_out\_sample\_idc greater than 4 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore SEI messages that contain reserved values of nnpfc\_out\_sample\_idc.

**nnpfc\_out\_tensor\_bitdepth\_minus8** plus 8 specifies the bit depth of sample values in the output integer tensor. The value of outTensorBitDepth is derived as follows:

outTensorBitDepth = nnpfc\_out\_tensor\_bitdepth\_minus8 + 8(84)

It is a requirement of bitstream conformance that the value of nnpfc\_out\_tensor\_bitdepth\_minus8 shall be in the range of 0 to 24, inclusive.

**nnpfc\_out\_order\_idc** indicates the output order of samples resulting from the post-processing filter. Table 24 contains an informative description of nnpfc\_out\_order\_idc values. The semantics of nnpfc\_out\_order\_idc in the range of 0 to 3, inclusive, are specified in Table 25, which specifies a process for deriving sample values in the filtered output sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic from the output tensors outputTensor for different values of nnpfc\_out\_order\_idc and a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensors. When nnpfc\_purpose is equal to 2 or 4, nnpfc\_out\_order\_idc shall not be equal to 3. The value of nnpfc\_out\_order\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_out\_order\_idc greater than 3 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore SEI messages that contain reserved values of nnpfc\_out\_order\_idc.

**Table 24 – Informative description of nnpfc\_out\_order\_idc values**

|  |  |
| --- | --- |
| **nnpfc\_out\_ order\_idc** | **Description** |
| 0 | Only the luma matrix is present in the output tensor, thus the number of channels is 1. |
| 1 | Only the chroma matrices are present in the output tensor, thus the number of channels is 2. |
| 2 | The luma and chroma matrices are present in the output tensor, thus the number of channels is 3. |
| 3 | Four luma matrices and two chroma matrices are present in the output tensor, thus the number of channels is 6. This nnpfc\_out\_order\_idc can only be used when the chroma format is 4:2:0. |
| 4..255 | reserved |

**Table 25 – Process for deriving sample values in the filtered output sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic from the output tensors outputTensor for a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensors**

|  |  |
| --- | --- |
| **nnpfc\_out\_ order\_idc** | **Process StoreOutputTensors( ) for deriving sample values in the filtered picture from the output tensors** |
| 0 | for( yP = 0; yP < outPatchHeight; yP++)  for( xP = 0; xP < outPatchWidth; xP++ ) {  yY = cTop \* outPatchHeight / inpPatchHeight + yP  xY = cLeft \* outPatchWidth / inpPatchWidth + xP  if ( yY < nnpfc\_pic\_height\_in\_luma\_samples && xY < nnpfc\_pic\_width\_in\_luma\_samples )  if( nnpfc\_component\_last\_flag = = 0 )  FilteredYPic[yY ][ xY ] = OutY( outputTensor[ 0 ][ 0 ][ yP ][ xP ] )  else  FilteredYPic[ yY ][ xY ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 0 ] )  } |
| 1 | for( yP = 0; yP < outPatchCHeight; yP++)  for( xP = 0; xP < outPatchCWidth; xP++ ) {  xSrc = cLeft \* horCScaling + xP  ySrc = cTop \* verCScaling + yP  if ( ySrc < nnpfc\_pic\_height\_in\_luma\_samples / outSubHeightC &&  xSrc < nnpfc\_pic\_width\_in\_luma\_samples / outSubWidthC )  if( nnpfc\_component\_last\_flag = = 0 ) {  FilteredCbPic[ ySrc ][ xSrc ] = OutC( outputTensor[ 0 ][ 0 ][ yP ][ xP ] )  FilteredCrPic[ ySrc ][ xSrc ] = OutC( outputTensor[ 0 ][ 1 ][ yP ][ xP ] )  } else {  FilteredCbPic[ ySrc ][ xSrc ] = OutC( outputTensor[ 0 ][ yP ][ xP ][ 0 ] )  FilteredCrPic[ ySrc ][ xSrc ] = OutC( outputTensor[ 0 ][ yP ][ xP ][ 1 ] )  }  } |
| 2 | for( yP = 0; yP < outPatchHeight; yP++)  for( xP = 0; xP < outPatchWidth; xP++ ) {  yY = cTop \* outPatchHeight / inpPatchHeight + yP  xY = cLeft \* outPatchWidth / inpPatchWidth + xP  yC = yY / outSubHeightC   xC = xY / outSubWidthC   yPc = ( yP / outSubHeightC ) \* outSubHeightC  xPc = ( xP / outSubWidthC ) \* outSubWidthC  if ( yY < nnpfc\_pic\_height\_in\_luma\_samples && xY < nnpfc\_pic\_width\_in\_luma\_samples)  if( nnpfc\_component\_last\_flag = = 0 ) {  FilteredYPic[ yY ][ xY ] = OutY( outputTensor[ 0 ][ 0 ][ yP ][ xP ] )  FilteredCbPic[ yC ][ xC ] = OutC( outputTensor[ 0 ][ 1 ][ yPc ][ xPc ] )  FilteredCrPic[ yC ][ xC ] = OutC( outputTensor[ 0 ][ 2 ][ yPc ][ xPc ] )  } else {  FilteredYPic[ yY ][ xY ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 0 ] )  FilteredCbPic[ yC ][ xC ] = OutC( outputTensor[ 0 ][ yPc ][ xPc ][ 1 ] )  FilteredCrPic[ yC ][ xC ] = OutC( outputTensor[ 0 ][ yPc ][ xPc ][ 2 ] )  }  } |
| 3 | for( yP = 0; yP < outPatchHeight; yP++ )  for( xP = 0; xP < outPatchWidth; xP++ ) {  ySrc = cTop / 2 \* outPatchHeight / inpPatchHeight + yP  xSrc = cLeft / 2 \* outPatchWidth / inpPatchWidth + xP  if ( ySrc < nnpfc\_pic\_height\_in\_luma\_samples / 2 &&  xSrc < nnpfc\_pic\_width\_in\_luma\_samples / 2 )  if( nnpfc\_component\_last\_flag = = 0 ) {  FilteredYPic[ ySrc \* 2 ][ xSrc \* 2 ] = OutY( outputTensor[ 0 ][ 0 ][ yP ][ xP ] )  FilteredYPic[ ySrc \* 2 ][ xSrc \* 2 + 1 ] = OutY( outputTensor[ 0 ][ 1 ][ yP ][ xP ] )  FilteredYPic[ ySrc \* 2 + 1 ][ xSrc \* 2 ] = OutY( outputTensor[ 0 ][ 2 ][ yP ][ xP ] )  FilteredYPic[ ySrc \* 2 + 1 ][ xSrc \* 2 + 1] = OutY( outputTensor[ 0 ][ 3 ][ yP ][ xP ] )  FilteredCbPic[ ySrc ][ xSrc ] = OutC( outputTensor[ 0 ][ 4 ][ yP ][ xP ] )  FilteredCrPic[ ySrc ][ xSrc ] = OutC( outputTensor[ 0 ][ 5 ][ yP ][ xP ] )  } else {  FilteredYPic[ ySrc \* 2 ][ xSrc \* 2 ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 0 ] )  FilteredYPic[ ySrc \* 2 ][ xSrc \* 2 + 1 ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 1 ] )  FilteredYPic[ ySrc \* 2 + 1 ][ xSrc \* 2 ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 2 ] )  FilteredYPic[ ySrc \* 2 + 1 ][ xSrc \* 2 + 1] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 3 ] )  FilteredCbPic[ ySrc ][ xSrc ] = OutC( outputTensor[ 0 ][ yP ][ xP ][ 4 ] )  FilteredCrPic[ ySrc ][ xSrc ] = OutC( outputTensor[ 0 ][ yP ][ xP ][ 5 ] )  }  } |
| 4..255 | reserved |

A base post-processing filter for a cropped decoded output picture picA is the filter that is identified by the first neural-network post-filter characteristics SEI message, in decoding order, that has a particular nnpfc\_id value within a CLVS.

If there is another neural-network post-filter characteristics SEI message that has the same nnpfc\_id value, has nnpfc\_mode\_idc equal to 1, has different content than the neural-network post-filter characteristics SEI message that defines the base post-processing filter, and pertains to the picture picA, the base post-processing filter is updated by decoding the ISO/IEC 15938-17 bitstream in that neural-network post-filter characteristics SEI message to obtain a post-processing filter PostProcessingFilter( ). Otherwise, the post-processing processing filter PostProcessingFilter( ) is assigned to be the same as the base post-processing filter.

The following process is used to filter the cropped decoded output picture with the post-processing filter PostProcessingFilter( ) to generate the filtered picture, which contains Y, Cb, and Cr sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic, respectively, as indicated by nnpfc\_out\_order\_idc.

if( nnpfc\_inp\_order\_idc = = 0 )  
 for( cTop = 0; cTop < InpPicHeightInLumaSamples; cTop += inpPatchHeight )  
 for( cLeft = 0; cLeft < InpPicWidthInLumaSamples; cLeft += inpPatchWidth ) {  
 DeriveInputTensors( )  
 outputTensor = PostProcessingFilter( inputTensor )  
 StoreOutputTensors( )  
 }  
else if( nnpfc\_inp\_order\_idc = = 1 )  
 for( cTop = 0; cTop < InpPicHeightInLumaSamples / InpSubHeightC; cTop += inpPatchHeight )  
 for( cLeft = 0; cLeft < InpPicWidthInLumaSamples / InpSubWidthC; cLeft += inpPatchWidth ) {  
 DeriveInputTensors( )  
 outputTensor = PostProcessingFilter( inputTensor )  
 StoreOutputTensors( )  
 }  
else if( nnpfc\_inp\_order\_idc = = 2 )  
 for( cTop = 0; cTop < InpPicHeightInLumaSamples; cTop += inpPatchHeight) (85)  
 for( cLeft = 0; cLeft < InpPicWidthInLumaSamples; cLeft += inpPatchWidth) {  
 DeriveInputTensors( )  
 outputTensor = PostProcessingFilter( inputTensor )  
 StoreOutputTensors( )  
 }  
else if( nnpfc\_inp\_order\_idc = = 3 )  
 for( cTop = 0; cTop < InpPicHeightInLumaSamples; cTop += inpPatchHeight \* 2 )  
 for( cLeft = 0; cLeft < InpPicWidthInLumaSamples; cLeft += inpPatchWidth \* 2 ) {  
 DeriveInputTensors( )  
 outputTensor = PostProcessingFilter( inputTensor )  
 StoreOutputTensors( )  
 }

**nnpfc\_reserved\_zero\_bit** shall be equal to 0.

**nnpfc\_payload\_byte**[ i ] contains the i-th byte of a bitstream conforming to ISO/IEC 15938-17. The byte sequence nnpfc\_payload\_byte[ i ] for all present values of i shall be a complete bitstream that conforms to ISO/IEC 15938-17.

**nnpfc\_parameter\_type\_flag** equal to 0 indicates that the neural network uses only integer parameters. nnpfc\_parameter\_type\_flag equal to 1 indicates that the neural network may use floating point or integer parameters.

**nnpfc\_log2\_parameter\_bit\_length\_minus3** equal to 0, 1, 2, and 3 indicates that the neural network does not use parameters of bit length greater than 8, 16, 32, and 64, respectively.

**nnpfc\_num\_parameters\_idc** indicates the maximum number of neural network parameters for the post processing filter in units of a power of 2048. nnpfc\_num\_parameters\_idc equal to 0 indicates that the maximum number of neural network parameters is not specified.

If the value of nnpfc\_num\_parameters\_idc is greater than zero, the variable maxNumParameters is derived as follows:

maxNumParameters = ( 2048  <<  nnpfc\_num\_parameters\_idc ) − 1(86)

It is a requirement of bitstream conformance that the number of neural network parameters of the post-processing filter shall be less than or equal to maxNumParameters.

**nnpfc\_num\_kmac\_operations\_idc** greater than 0 specifies that the maximum number of multiply-accumulate operations per sample of the post-processing filter is less than or equal to nnpfc\_num\_kmac\_operations\_idc \* 1000. nnpfc\_num\_kmac\_operations\_idc equal to 0 specifies that the maximum number of multiply-accumulate operations of the network is not specified.

* 1. **Neural-network post-filter activation SEI message**
     1. Neural-network post-filter activation SEI message syntax

|  |  |
| --- | --- |
| nn\_post\_filter\_activation( payloadSize ) { | **Descriptor** |
| **nnpfa\_id** | ue(v) |
| } |  |

* + 1. Neural-network post-filter activation SEI message syntax

This SEI message specifies the neural-network post-processing filter that may be used for post-processing filtering for the current picture.

The neural-network post-processing filter activation SEI message persists only for the current picture.

NOTE – There may be several neural-network post-processing filter activation SEI messages present for the same picture, for example, when the post-processing filters are meant for different purposes or filter different colour components.

**nnpfa\_id** specifies that the neural-network post-processing filter specified by one or more neural-network post-processing filter characteristics SEI messages that pertain to the current picture and have nnpfc\_id equal to nnfpa\_id may be used for post-processing filtering for the current picture.

*Add the following to the Bibliography and renumber the bibliographic references as needed:*

ATSC A/341:2022-03, *ATSC Standard: Video – HEVC*.

**Changes to the VVC specification text:**

[Ed. (MH): Move the text below to a VVC amendment later. For now, the VVC text related to the SEI messages specified in this document is maintained below in order to avoid losing it completely.]

*In clause D.2.1 of VVC, add the following, highlighted table rows:*

|  |  |
| --- | --- |
| sei\_payload( payloadType, payloadSize ) { | **Descriptor** |
| if( nal\_unit\_type = = PREFIX\_SEI\_NUT ) |  |
| ... |  |
| else if( payloadType = = 209 ) /\* Specified in Rec. ITU-T H.274 | ISO/IEC 23002-7 \*/ |  |
| shutter\_interval\_info( payloadSize ) |  |
| else if( payloadType = = 210 ) /\* Specified in Rec. ITU-T H.274 | ISO/IEC 23002-7 \*/ |  |
| nn\_post\_filter\_characteristics( payloadSize ) |  |
| else if( payloadType = = 211 ) /\* Specified in Rec. ITU-T H.274 | ISO/IEC 23002-7 \*/ |  |
| nn\_post\_filter\_activation( payloadSize ) |  |
| else /\* Specified in Rec. ITU-T H.274 | ISO/IEC 23002-7 \*/ |  |
| reserved\_message( payloadSize ) |  |
| ... |  |

*In clause D.2.2 of VVC, replace the following:*

The list VclAssociatedSeiList is set to consist of the payloadType values 3, 19, 45, 47, 129, 132, 137, 142, 144, 145, 147 to 150, inclusive, 153 to 156, inclusive, 165, 168, 177, 179, 180, 200 to 202, inclusive, and 204 to 207, inclusive.

*with:*

The list VclAssociatedSeiList is set to consist of the payloadType values 3, 19, 45, 47, 129, 132, 137, 142, 144, 145, 147 to 150, inclusive, 153 to 156, inclusive, 165, 168, 177, 179, 180, 200 to 202, inclusive, 204 to 207, inclusive, and 209 to 211, inclusive.

*Add clause D.14.10 and D.14.11 of VVC as follows:*

**D.14.10 Use of the shutter interval information information SEI message**

The following constraints apply to the shutter interval information SEI message:

– When the value of SpsMaxSubLayersMinus1 is equal to 0, the value of fixed\_shutter\_interval\_within\_clvs\_flag shall be equal to 1.

– The value of sii\_max\_sub\_layers\_minus1 shall be equal to the value of SpsMaxSubLayersMinus1.

**D.14.11 Use of the neural network post-filter characteristics SEI message**

For purposes of interpretation of the neural-network post-filter characteristics SEI message, the following variables are specified:

– InpPicWidthInLumaSamples is set equal to pps\_pic\_width\_in\_luma\_samples − ‌SubWidthC \* ( pps\_conf\_win\_left\_offset + pps\_conf\_win\_right\_offset ).

– InpPicHeightInLumaSamples is set equal to pps\_pic\_height\_in\_luma\_samples − ‌SubHeightC \* ( pps\_conf\_win\_top\_offset + pps\_conf\_win\_bottom\_offset ).

– The variables CroppedYPic[ y ][ x ] and chroma sample arrays CroppedCbPic[ y ][ x ] and CroppedCrPic[ y ][ x ], when present, are set to be the 2-dimensional arrays of decoded sample values of the 0th, 1st, and 2nd component, respectively, of the cropped decoded output picture to which the neural-network post-filter characteristics SEI message applies.

– BitDepthY and BitDepthC are both set equal to BitDepth.

– InpSubWidthC is set equal to SubWidthC.

– InpSubHeightC is set equal to SubHeightC.

– SliceQPY is set equal to SliceQpY.

When a neural-network post-filter characteristics SEI message with the same nnpfc\_id and different content are present in the same picture unit, both neural-network post-filter characteristics SEI messages shall be present in the same SEI NAL unit.